

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Inmplode: A Framework to Interpret Multiple Related Rule-Based Models

Pedro Rodrigo Caetano Strecht Ribeiro

THESIS PRESENTED IN PARTIAL FULFILLMENT OF REQUIREMENTS
FOR THE DEGREE OF DOCTOR IN INFORMATICS ENGINEERING



Programa Doutoral em Engenharia Informática

Supervisors:

João Pedro Carvalho Leal Mendes Moreira
Carlos Manuel Milheiro de Oliveira Pinto Soares

2025

Inmplode: A Framework to Interpret Multiple Related Rule-Based Models

Pedro Rodrigo Caetano Strecht Ribeiro

Programa Doutoral em Engenharia Informática

Abstract

This thesis investigates the challenges and opportunities presented by the increasing trend of using multiple specialized models, referred to as *operational models*, to address complex data analysis problems. While such an approach can enhance predictive performance for specific sub-problems, it often leads to fragmented knowledge and difficulties understanding overarching organizational phenomena. This research focuses on synthesizing the knowledge embedded within a collection of decision tree models chosen for their inherent interpretability and suitability for knowledge extraction. For example, a company with chain stores or a university with diverse programs, each using dedicated prediction models (sales or dropout, respectively). While these localized models are important, a global perspective is valuable organization-wide. However, managing many operational models, especially for cross-program/store analysis, can be overwhelming.

A methodology framed within a comprehensive framework is introduced to merge sets of operational models into *consensus models*. These consensus models are directed towards higher-level decision-makers, enhancing the interpretability of knowledge generated by the operational models. The framework, named *Inmplode*, addresses common challenges in model merging and presents a highly customizable process. This process features a generic workflow and adaptable components, detailing alternative approaches for each subproblem encountered in the merging process.

The framework was applied to four public datasets from diverse business areas and a case study in education using data from the University of Porto. Different model merging approaches were explored in each case, illustrating various process instantiations. The model merging process revealed that the resulting consensus models are frequently incomplete, meaning they cannot cover the entire decision space, which can undermine their intended purpose. To address the issue of incompleteness, two novel methodologies are explored: one relies on the generation of synthetic datasets followed by decision tree training. At the same time, the other uses a specialized algorithm designed to construct a decision tree directly from aggregated (i.e., symbolic) data. The effectiveness of these methodologies in generating complete consensus models from incomplete rule sets is evaluated across the five datasets. Empirical results demonstrate the feasibility of overcoming the incompleteness issue, contributing to knowledge synthesis and decision tree modeling. However, tradeoffs were identified between completeness and interpretability, predictive performance, and the fidelity of consensus models.

Overall, this research addresses a critical gap in the literature by providing a comprehensive framework for synthesizing knowledge from multiple decision tree models, focusing on overcoming the challenge of incompleteness. The conclusions have implications for organizations seeking to use specialized models while maintaining a holistic understanding of the analyzed phenomenon.

Resumo

Esta tese investiga os desafios e as oportunidades apresentados pela tendência crescente de utilização de múltiplos modelos especializados, designados por *modelos operacionais*, para abordar problemas complexos de análise de dados. Embora esta abordagem possa melhorar o desempenho preditivo para subproblemas específicos, frequentemente conduz a um conhecimento fragmentado e a dificuldades na compreensão de fenómenos a nível de uma organização. Esta investigação centra-se na síntese do conhecimento incorporado num conjunto de modelos de árvores de decisão, escolhidos pela sua interpretabilidade inerente e adequação à extração de conhecimento. Por exemplo, uma empresa com cadeias de lojas ou uma universidade com diversos cursos, cada uma utilizando modelos de previsão dedicados (volume de vendas ou abandono escolar, respetivamente). Embora estes modelos localizados sejam importantes, uma perspetiva global é importante a nível organizacional. Contudo, a gestão de muitos modelos operacionais, especialmente para análises interprogramas/lojas, pode ser complexa.

É introduzida uma *framework* para fundir conjuntos de modelos operacionais em *modelos consensuais*. Estes modelos são direcionados a decisores de níveis superiores, melhorando a interpretabilidade do conhecimento gerado pelos modelos operacionais. A *framework*, denominada *Inmplode*, aborda desafios comuns na agregação de modelos e apresenta um processo customizável com um fluxo de trabalho genérico e componentes adaptáveis, detalhando abordagens alternativas para cada subproblema encontrado no processo de agregação.

A *framework* foi aplicada a quatro conjuntos de dados públicos de diversas áreas de negócio e a um estudo de caso na educação com dados da Universidade do Porto. Em cada caso, foram exploradas diferentes abordagens de agregação de modelos, ilustrando várias instanciações do processo. O processo de agregação de modelos revelou que os modelos consensuais resultantes são frequentemente incompletos, ou seja, não conseguem cobrir todo o espaço de decisão, o que pode comprometer o seu propósito. Para abordar o prolema da incompletude, são exploradas duas novas metodologias: uma baseia-se na geração de conjuntos de dados sintéticos seguida de treino de árvores de decisão, enquanto a outra utiliza um algoritmo especializado concebido para construir uma árvore de decisão diretamente a partir de dados agregados (i.e., simbólicos). A eficácia destas metodologias na geração de modelos consensuais completos a partir de conjuntos de regras incompletos é avaliada nos cinco conjuntos de dados. Os resultados empíricos demonstram a viabilidade de superar a incompletude, constituindo um contributo para a área da síntese de conhecimento e modelação com árvores de decisão. Contudo, foram identificados compromissos entre a completude e a interpretabilidade, bem como entre o desempenho preditivo e a fidelidade dos modelos consensuais.

Globalmente, esta investigação aborda uma lacuna crítica na literatura, fornecendo um *framework* para sintetizar conhecimento a partir de múltiplos modelos de árvores de decisão, com particular foco no desafio da incompletude. As conclusões têm implicações significativas para organizações que procuram usar modelos especializados, obtendo também uma compreensão holística do fenómeno analisado.

Acknowledgments

Completing this thesis marks the culmination of a challenging and deeply rewarding journey, one that would not have been possible without the help and support of many people.

First and foremost, I wish to thank my supervisors, João Mendes-Moreira and Carlos Soares. They have been an unparalleled source of guidance, expertise, resources, and encouragement throughout this research. They were the ideal supervising team for me, offering both the freedom to explore and invaluable advice on navigating the complexities of this field. They were also instrumental in helping me refine my scientific writing skills and meticulously reviewing my work. I am deeply grateful for their patience and unwavering support, for this long and, at times, slow-paced, journey.

I am also thankful to co-authors, Luís Cruz and Rui Abreu. Together with my supervisors, we collaborated on a paper that has made some impact in the field of Educational Data Mining, where this research initially emerged.

I extend my gratitude to António Augusto Sousa, Eugénio Oliveira, and Eduarda Mendes Rodrigues from the Doctoral Program in Informatics Engineering at FEUP, who were an early source of inspiration and provided foundational knowledge of research methodologies. I am also grateful to my colleagues in ProDEI for the mutual support we shared as we embarked on this path together.

This thesis has led to a number of publications, supported by various research missions. I am grateful for the funding provided by the Department of Informatics Engineering at FEUP and INESC TEC (LIADD), which made these enriching scientific experiences possible. During these travels, I particularly value the warm reception I received at NTNU from Gry Jordet Kibsgaard and Torgeir Sesseng. I also thank Elsa Cardoso at ISCTE and look forward to continuing our collaboration in the future.

I wish to express my gratitude to Lúcia Ribeiro and Tito Vieira for their early support and recommendations from the very beginning of this journey. I also thank the Rectory of the University of Porto for granting access to the University's data. I am thankful to Elisabete Neves, Fernando Silva, and Ana Camanho at UPdigital for the opportunity to apply my machine learning expertise on a project focused on predicting student dropout. I am also thankful to Vítor Carvalho and Carla Amaral for their help with the graphical materials for this document. To my friends and colleagues at UPdigital, I must take this opportunity to highlight what an incredible team we are!

To all my friends, both recent and long-standing, it would be impossible to name you all here, but you know who you are and how much you mean to me. I look forward to celebrating this milestone with all of you soon!

Finally, I would like to express my deepest gratitude to my family and loved ones for their unconditional love and support throughout this journey.

Pedro Strecht

“Algorithms are an exacting standard. It’s often said that you don’t really understand something until you can express it as an algorithm.”

Pedro Domingos

Contents

Abstract	i
Acknowledgments	v
Contents	xii
List of Figures	xv
List of Tables	xix
List of Algorithms	xxi
Listings	xxiii
Symbols	xxv
1 Introduction	1
1.1 Multiple models for related tasks: the case of predicting academic success	2
1.1.1 Decision levels in hierarchical organizations	2
1.1.2 Separate models for related problems	3
1.2 Consensus models as a holistic representation of knowledge	5
1.3 Hypothesis and approach	6
1.4 Contributions and publications	8
1.4.1 Main contributions	8
1.4.2 Additional contributions	9
1.5 Thesis outline	10
2 Background	13
2.1 Machine learning essential concepts and fundamentals	13
2.1.1 Knowledge discovery, data mining, and machine learning	13
2.1.2 Data representation and related concepts	15
2.1.3 Training of prediction models	17
2.1.4 Evaluation of predictions models performance	19
2.2 Rule-based models and related formats	22
2.2.1 Decision trees	25
2.2.2 Binary decision diagrams	27
2.2.3 Classification rules	28
2.2.4 Decision tables	30
2.3 Summary	32

CONTENTS

3	Related Work	33
3.1	Model interpretability	33
3.1.1	Interpretability and explainability	35
3.1.2	Desiderata of interpretable models	36
3.1.3	Interpretability taxonomy overview	37
3.1.4	Evaluation of interpretability	39
3.1.5	Interpretability challenges in rule-based models	49
3.2	Methods to merge prediction models	51
3.2.1	Combining models: ensemble vs. merging methods	51
3.2.2	Analytical approaches to merge models	53
3.2.3	Mathematical approaches to merge models	56
3.3	Summary	57
4	<i>Inmplode</i>: A Framework to Interpret Multiple Related Rule-based Models	59
4.1	Model merging in decentralized organizations	59
4.2	The <i>Inmplode</i> framework	60
4.2.1	Generic workflow	61
4.2.2	Customizable components	62
4.2.3	Formal overview	63
4.3	Creation of folds for each dataset	67
4.4	Training of operational models	69
4.5	Evaluation of operational models	70
4.5.1	Evaluation of performance	72
4.5.2	Evaluation of interpretability	76
4.6	Stratification of operational models	76
4.7	Merging operational models into a consensus model	80
4.8	Evaluation of consensus models	81
4.8.1	Evaluation of performance	83
4.8.2	Evaluation of performance loss	83
4.8.3	Evaluation of fidelity	85
4.9	Collection of results	88
4.10	Summary	90
5	Merging a Set of Rule-based Models into a Single Model	91
5.1	Merging a set of models into a single model	91
5.2	Converting a model to a decision table	93
5.3	Expanded rule representation of a decision table	95
5.4	Merging two decision tables	99
5.4.1	Merging rules sets	100
5.4.2	Merging a pair of rules	101
5.4.3	Intersection of rules	101
5.4.4	Estimation of the evaluation measures of a new rule	105
5.5	Building a merged decision table	106
5.6	Handling rule's conflicts	109
5.7	Minimizing the number of rules	112
5.8	Specific merging scenarios	116
5.9	Converting a decision table to a model	119
5.10	Summary	120

CONTENTS

6	Methods to Complete Consensus Rule-based Models	121
6.1	Training decision trees from incomplete decision tables	121
6.2	Synthetic data trees method	123
6.2.1	Generate synthetic data from a decision table	123
6.2.2	Train a model from synthetic data	126
6.3	Distributional data trees method	127
6.3.1	The Symbolic Data Analysis paradigm	127
6.3.2	Decision tables as symbolic data	131
6.3.3	Generate distributional data from a decision table	131
6.3.4	Train a model from distributional data	135
6.4	Limitations in decision tree generation from decision tables	139
6.5	Summary	140
7	Empirical Evaluation of <i>Inmplode</i>	141
7.1	Experimental setup	141
7.1.1	Selected datasets	141
7.1.2	Evaluation methodology	142
7.2	Results of empirical evaluation	144
7.2.1	Predictive performance	144
7.2.2	Predictive performance loss	146
7.2.3	Fidelity	149
7.2.4	Interpretability	152
7.2.5	Completeness	160
7.3	Correlation analysis of evaluation metrics	167
7.3.1	Predictive performance metrics	169
7.3.2	Predictive performance and Fidelity	170
7.3.3	Predictive performance and Interpretability	170
7.3.4	Predictive performance and Completeness	171
7.3.5	Fidelity metrics	171
7.3.6	Fidelity and Interpretability	172
7.3.7	Fidelity and Completeness	172
7.3.8	Interpretability metrics	173
7.3.9	Interpretability and Completeness	174
7.3.10	Completeness metrics	175
7.4	Key trends and trade-offs from consensus model evaluation	176
7.5	Summary	177
8	Conclusions	179
8.1	Summary and contributions	179
8.2	Evaluation of the hypothesis	181
8.3	Future work	182
A	Density Estimation in High Dimensional Spaces: a Multivariate Histogram Approach	183
A.1	Density estimation and related concepts	183
A.1.1	Approaches to density estimation	184
A.1.2	Applications in research	187
A.1.3	Example: Density of the Old Faithful dataset	187
A.2	A multivariate histogram-based method for density estimation	188
A.2.1	Empirical domains	190

CONTENTS

A.2.2	Hypergrid	191
A.2.3	Multivariate density estimator	193
A.3	Handling datasets with missing data	194
A.4	Evaluation on selected datasets	197
A.4.1	Evaluation of computational performance in density estimation	198
A.4.2	Evaluation of the impact of categorical variables in density estimation	201
A.4.3	Evaluation of the impact of missing data in density estimation	204
A.5	Summary	204
B	Measuring Completeness: An Evaluation for Rule-Based Models	207
B.1	Completeness in machine learning	207
B.1.1	Completeness in data	207
B.1.2	Completeness in rule-based models	208
B.1.3	Completeness in fuzzy expert systems	209
B.1.4	Completeness in interpretability	210
B.1.5	Completeness in performance evaluation	210
B.2	A taxonomy of decision spaces	211
B.3	Completeness of rule-based models	213
B.3.1	Example: Incompleteness as a result of model merging	214
B.3.2	Example: Completeness in the context of decision spaces	216
B.4	Metrics to measure completeness of rule-based models	217
B.4.1	Geometrical completeness	218
B.4.2	Distributional completeness	220
B.5	Evaluation on selected datasets	221
B.5.1	Experimental setup	221
B.5.2	Results and discussion	222
B.6	Summary	227
C	Datasets, Models and Evaluation Results	231
C.1	Datasets description	231
C.1.1	Adult Census dataset	232
C.1.2	Bank Marketing dataset	233
C.1.3	Hotel Reservations dataset	233
C.1.4	Flight Delays dataset	233
C.1.5	University of Porto dataset	234
C.2	Business units and operational models	235
C.2.1	Examples of operational models	236
C.3	Strata and consensus models	236
C.3.1	Examples of consensus models	239
C.4	Detailed results of evaluation	239
C.4.1	Results of evaluation of Predictive performance	239
C.4.2	Results of evaluation of Predictive performance loss	247
C.4.3	Results of evaluation of Fidelity	253
C.4.4	Results of evaluation of Interpretability	259
C.4.5	Results of evaluation of Completeness	270
	References	281

List of Figures

1.1	Decision-makers across the organizational units of a higher education institution .	3
1.2	Creating professor/lecturer level models	4
1.3	Creating a single decision level model by dataset concatenation	4
1.4	Merging operational models into a consensus model	5
2.1	The Knowledge Discovery in Databases process	15
2.2	Estimation of performance with the holdout method	21
2.3	Estimation of performance with the k -fold cross-validation method	23
2.4	A colloquial description of a rule-based model	25
2.5	A rule-based model represented as a binary decision tree	26
2.6	A rule-based model represented as a binary decision diagram	28
2.7	A rule-based model represented as a decision set	29
2.8	A rule-based model represented as a simplified decision set	30
2.9	A rule-based model represented as a decision set in the canonical form	30
3.1	Black-box model	34
3.2	White-box model	34
3.3	Interpretability taxonomy overview	38
3.4	Standard approach to fidelity evaluation	48
3.5	Ensemble method setup	52
3.6	Merging method setup	53
3.7	Creating a model by moving datasets	54
3.8	Creating a new model by moving and merging models	54
3.9	Merging models to create a model from a large dataset	55
4.1	The <i>Inmplode</i> framework generic workflow	62
4.2	Creation of K folds for N datasets	68
4.3	Training an operational model for deployment	69
4.4	Training and evaluation of operational models	71
4.5	Model performance across different setups in the case study	75
4.6	Variable importance distributions across different setups in the case study	75
4.7	Stratification of operational models into multiple strata	78
4.8	Models contained in groups across different grouping criteria (setup #1)	81
4.9	Setups for evaluating the predictive performance of a consensus model for stratum j	84
4.10	Setups for evaluating the fidelity of a consensus model for stratum j	86
5.1	Merging a set of models into a single model	92
5.2	A decision tree model and underlying decision regions	94
5.3	Decision regions derived from a decision tree	95

LIST OF FIGURES

5.4	A single leaf decision tree	95
5.5	A single rule decision table	95
5.6	Geometrical depiction of merging two models by intersection	104
6.1	Uncovered region in a decision space	122
6.2	Synthetic data trees method	123
6.3	Taxonomy of single-valued and symbolic variables	129
6.4	Distributional data trees method	132
7.1	Results of the Predictive performance of consensus models measured with MCC .	147
7.2	Results of the Predictive performance of operational models measured with MCC	150
7.3	Results of the Predictive performance loss of consensus models measured with ΔMCC	151
7.4	Results of the Fidelity of consensus models measured with FidMCC	153
7.5	Results of the Interpretability of operational models measured with Size	156
7.6	Results of the Interpretability of operational models measured with Dimensionality	157
7.7	Results of the Interpretability of operational models measured with Depth	158
7.8	Results of the Interpretability of consensus models measured with Size	159
7.9	Results of the Interpretability of consensus models measured with Dimensionality	160
7.10	Results of the Interpretability of consensus models measured with Depth	161
7.11	Results of the Completeness of consensus models measured with Fraction covered	163
7.12	Results of the Completeness of consensus models measured with Geometrical completeness	165
7.13	Results of the Completeness of consensus models measured with Distributional completeness	166
7.14	Correlation matrix of evaluation metrics for Merged tables	168
7.15	Correlation matrix of evaluation metrics for Synthetic data trees	168
7.16	Correlation matrix of evaluation metrics for Distributional data trees	169
7.17	Correlations between Predictive performance (MCC) and Fidelity (FidMCC) . .	170
7.18	Correlations between Predictive performance (MCC) and Completeness (DComp)	171
7.19	Correlations between Fidelity (FidMCC) and Interpretability (Dimensionality) . .	172
7.20	Correlations between Fidelity (FidMCC) and Completeness (DComp)	173
7.21	Correlations between Interpretability metrics (Size and Dimensionality)	174
7.22	Correlations between Interpretability (Size) and Completeness (DComp)	175
7.23	Correlations between Interpretability (Dimensionality) and Completeness (DComp)	175
7.24	Correlations between Completeness metrics (GComp and DComp)	176
A.1	Old Faithful data	188
A.2	Density of ‘waiting’	188
A.3	Density of ‘duration’	188
A.4	10×10 histogram	189
A.5	30×30 histogram	189
A.6	Density surface	189
A.7	Overview of the multivariate histogram-based approach	190
A.8	A 3-D hypergrid with 18 hyperrectangles	192
A.9	Subtasks to handle datasets with missing data	195
A.10	Undefined hyperrectangle	196
A.11	Missing values 2-D hypergrid	196
A.12	Complete hyperrectangles	196

LIST OF FIGURES

A.13 Smaller datasets density estimation computational time ($\mu < 30000$)	201
A.14 Larger datasets density estimation computational time ($\mu \geq 30000$)	201
B.1 A taxonomy of decision spaces	213
B.2 Geometric depiction of model intersection coverage	215
B.3 An attribute space, an instance space, and a learned space	217
B.4 A learned space within an instance space	219
B.5 A hypergrid as a basis for a density estimator	221
B.6 A learned space within a hypergrid	222
B.7 Evaluating completeness of merged decision tables	225
B.8 Adult Census dataset (setup #1): Completeness in merged decision tables	226
B.9 Bank Marketing dataset (setup #1): Completeness in merged decision tables . . .	226
B.10 Hotel Reservations dataset (setup #1): Completeness in merged decision table . .	226
B.11 Flight Delays dataset (setup #1): Completeness in merged decision tables	227
B.12 University of Porto dataset (setup #1): Completeness in merged decision tables .	227
B.13 Adult Census dataset: Completeness in merged decision tables in setups	228
B.14 Bank Marketing dataset: Completeness in merged decision tables in setups	228
B.15 Hotel Reservations dataset: Completeness in merged decision tables in setups . .	228
B.16 Flight Delays dataset: Completeness in merged decision tables in setups	229
B.17 University of Porto dataset: Completeness in merged decision tables in setups . .	229

LIST OF FIGURES

List of Tables

2.1	Structure of a dataset	17
2.2	Structure of a training set	18
2.3	Confusion matrix	19
2.4	A rule-based model represented as a single-hit decision table	31
2.5	Structure of an extended attributive decision table	32
2.6	A rule-based model represented as an extended attributive decision table	32
3.1	Rule confusion matrix	42
3.2	Confusion matrix to evaluate the agreement between two models	48
4.1	Datasets variables in the case study	67
4.2	Example of a model evaluation specifications set ($\mathcal{Z}$)	72
4.3	Example of strata created from stratification variable values	77
4.4	Models' dataset created in setup #1 in the case study	80
5.1	Decision table derived from a decision tree	95
5.2	Expanded rule representation in a decision table	97
5.3	A decision table extracted from a C5.0 decision tree model	99
5.4	Decision table T_A	106
5.5	Decision table T_B	106
5.6	Merging decision tables T_A and T_B	107
5.7	Building a merged table	109
5.8	Conflict resolution in merged table	112
5.9	Identifying the initial sets of rules to join (1st iteration – Table T_C)	116
5.10	Rules joined (1st iteration – Table T_D)	116
5.11	Identifying the next sets of rules to join (2nd iteration – Table T_C)	116
5.12	Rules joined (2nd iteration – Table T_D)	116
5.13	Decision table T_A (1st scenario)	117
5.14	Decision table T_B (1st scenario)	117
5.15	Rule-set of the intersection of the decision tables (1st scenario)	117
5.16	Merged decision table (1st scenario)	117
5.17	Decision table T_A (2nd scenario)	118
5.18	Decision table T_B (2nd scenario)	118
5.19	Rule-set of the intersection of the decision tables (2nd scenario)	118
5.20	Merged decision table (2nd scenario)	118
5.21	Decision table T_A (3rd scenario)	118
5.22	Decision table T_B (3rd scenario)	118
5.23	Rule-set of the intersection of the decision tables (3rd scenario)	119
5.24	Merged decision table (3rd scenario)	119

LIST OF TABLES

5.25	Decision table	119
6.1	Decision table resulting from merging six models (Bank marketing dataset) . . .	122
6.2	Methods for generating values from a subset of a hyperrectangle	126
6.3	A single-valued synthetic dataset (D_s)	127
6.4	Student enrollments data	129
6.5	Student enrollments variables	130
6.6	Course enrollments data	130
6.7	Course enrollments variables	130
6.8	Distribution of Contact in the Bank marketing Dataset	133
6.9	Distribution of Age in the Bank marketing Dataset	134
6.10	A probabilistic data table	137
6.11	Probabilities for questions within $t'_{4,3}$	138
6.12	Probabilities for questions within $t'_{1,1}$	138
7.1	Datasets used to evaluate consensus models developed with <i>Inmplode</i>	142
7.2	<i>Inmplode</i> configuration parameters in experiments	143
7.3	Dataset specific parameters settings	144
7.4	Model evaluation parameter settings ($\mathcal{Z}$)	144
A.1	Hypergrid configuration	192
A.2	Hypergrid definition	192
A.3	A density estimator based on a multivariate histogram ($\Gamma_{\hat{p}}$)	194
A.4	Density estimator intermediate calculations	195
A.5	Incomplete multivariate histogram ($\Gamma_f^{(1)}$)	197
A.6	Missing values multivariate histogram (Γ_{ϕ})	198
A.7	Completed multivariate histogram ($\Gamma_f^{(2)}$)	199
A.8	Coalesced multivariate histogram ($\Gamma_f^{(3)}$)	199
A.9	Datasets selected for the density estimation method evaluation	200
A.10	Density estimator without categorical variables ($d_c = 0$)	202
A.11	Density estimator with one categorical variable ($d_c = 1$)	203
A.12	Density estimator with two categorical variables ($d_c = 2$)	203
A.13	Density estimator with three categorical variables ($d_c = 3$)	203
A.14	Density estimator with controlled addition of missing data ($r_{mv} > 0$)	204
B.1	Multivariate density estimator ($\Gamma_{\hat{p}}$)	223
B.2	Multivariate density estimator restricted to the learned space ($\Gamma_{\hat{p}_M}$)	224
C.1	Datasets summary	231
C.2	Description of variables of the Adult Census dataset	232
C.3	Description of variables of the Bank Marketing dataset	233
C.4	Description of variables of the Hotel Reservations dataset	234
C.5	Description of variables of the Flight Delays dataset	234
C.6	Description of variables of the University of Porto dataset	235
C.7	Operational model development summary	236
C.8	Consensus model development summary	238
C.9	Predictive performance of consensus models from the Adult Census dataset . . .	241
C.10	Predictive performance of consensus models from the Bank Marketing dataset . .	242
C.11	Predictive performance of consensus models from the Hotel Reservations dataset	242

LIST OF TABLES

C.12 Predictive performance of consensus models from the Flight Delays dataset . . .	242
C.13 Predictive performance of consensus models from the University of Porto dataset	243
C.14 Predictive performance loss of consensus models from the Adult Census dataset .	247
C.15 Predictive performance loss of consensus models from the Bank Marketing dataset	247
C.16 Predictive performance loss of consensus models from the Hotel Reservations dataset	248
C.17 Predictive performance loss of consensus models from the Flight Delays dataset .	248
C.18 Predictive performance loss of consensus models from the University of Porto dataset	249
C.19 Fidelity of consensus models from the Adult Census dataset	253
C.20 Fidelity of consensus models from the Bank Marketing dataset	253
C.21 Fidelity of consensus models from the Hotel Reservations dataset	254
C.22 Fidelity of consensus models from the Flight Delays dataset	254
C.23 Fidelity of consensus models from the University of Porto dataset	255
C.24 Interpretability of consensus models from the Adult Census dataset	259
C.25 Interpretability of consensus models from the Bank Marketing dataset	259
C.26 Interpretability of consensus models from the Hotel Reservations dataset	260
C.27 Interpretability of consensus models from the Flight Delays dataset	260
C.28 Interpretability of consensus models from the University of Porto dataset	261
C.29 Completeness of consensus models from the Adult Census dataset	270
C.30 Completeness of consensus models from the Bank Marketing dataset	270
C.31 Completeness of consensus models from the Hotel Reservations dataset	271
C.32 Completeness of consensus models from the Flight Delays dataset	271
C.33 Completeness of consensus models from the University of Porto dataset	272

LIST OF TABLES

List of Algorithms

1	Inmplode: A framework for training, merging, and evaluating rule-based models	66
2	EvaluateOperationalModel: Evaluate an operational model from multiple criteria	72
3	EvaluatePerformance: Evaluate the predictive performance of a model	73
4	EvaluateInterpretability: Evaluate the interpretability of a rule-based model . . .	76
5	EvaluateConsensusModel: Evaluate a consensus model from multiple criteria . .	82
6	EvaluatePerformanceLoss: Evaluate the performance loss of a consensus model relative to a set of operational models	85
7	EvaluateConsensusFidelity: Evaluate the fidelity of a consensus model relative to a set of operational models	87
8	EvaluateFidelity: Evaluate the fidelity of a surrogate model relative to a reference model	88
9	AverageEvaluations: Averaging a model evaluation results across setups	90
10	MergeModels: Merge a set of rule-based models	93
11	MergeTables: Attempt to merge two decision tables	100
12	MergeRulesSets: Merge the rule sets of two decision tables	101
13	CreateRule: Create a rule by merging a pair of rules	102
14	IntersectRules (an instantiation of <i>MergePairRules</i>)	103
15	AverageMeasures (an instantiation of <i>EstimateRuleMeasures</i>)	105
16	BuildMergedTable: Evaluate a merge attempt of two decision tables	108
17	RejectBelowCoverage (an instantiation of <i>FilterRule</i>)	109
18	Resolve conflicts	111
19	RuleHighestCoverage (an instantiation of <i>ResolveRuleConflict</i>)	111
20	RuleGluing (an instantiation of <i>MinimizeRuleSet</i>)	114

LIST OF ALGORITHMS

Listings

2.1	A binary decision tree generated by the C5.0 algorithm	27
5.1	A decision tree model trained with the C5.0 algorithm	98
5.2	Decision tree from a decision table	119
6.1	Decision tree trained from synthetic data with the C5.0 algorithm	128
6.2	Decision tree trained from distributional data with the TREE algorithm	139
C.1	Operational model for country "France" (Adult Census dataset)	236
C.2	Operational model for month "August" (Bank Marketing dataset)	237
C.3	Operational model for month "11/2017" (Hotel Reservations dataset)	237
C.4	Operational model for airport "ABE" (Flight Delays dataset)	237
C.5	Operational model for course "18071" (University of Porto dataset)	238
C.6	Consensus model for world region "South America" (Adult Census dataset) . . .	240
C.7	Consensus model for semester "1st semester" (Bank Marketing dataset)	240
C.8	Consensus model for semeste "2017-2nd semester" (Hotel Reservations dataset) .	240
C.9	Consensus model for US region "Pacific" (Flight Delays dataset)	240
C.10	Consensus model for faculty "Sports" (University of Porto dataset)	241

Symbols

e	Example (instance, observation, sample, object, data point)
x	Variable (attribute, feature, dimension)
$v_{i,j}$	Value of variable x_j in example e_i
c	Category or number in a set of values
l	Lower bound in an interval
u	Upper bound in an interval
$\text{dom } x$	Domain of variable x
D	Dataset
n	Cardinality (size) of a dataset (number of examples)
X	Set of variables in a dataset
d	Dimensionality of a dataset (number of variables)
μ	Volume of a dataset ($n \times d$)
Tr	Training set
X_{Tr}	Set of independent variables
y	Target (or dependent) variable
V_{Tr}	Variables in a training set
M	Model
$\mathcal{L}$	Learning algorithm
$\hat{y}$	Prediction made by a model
Ts	Test set
$\oplus$	Positive class
P_a	Actual positives
N_a	Actual negatives
P_p	Predicted positives
N_p	Predicted negatives
TP	True positives
TN	True negatives
FN	False negatives
FP	False positives
PA	Positive agreements
NA	Negative agreements

SYMBOLS

PD	Positive disagreements
ND	Negative disagreements
f_i	Fold i of a dataset
r	Rule
p	Proposition (predicate, condition)
$t_{i,j}$	Subset of variable's x_j domain in rule r_i
s	Itemset of a rule
$ s $	Number of propositions in a rule
R	Number of rules in a rule-based model
$\mathcal{R}$	Rule-set of a rule-based model
B	Number of examples that activate a rule
$\bar{B}$	Number of examples that do not activate a rule
H	Number of examples where the predicted class matches the actual class
$\bar{H}$	Number of examples where the predicted class differs from the actual class
HB	Number of correctly predicted positive examples that are covered by a rule
$\bar{HB}$	Number of incorrectly predicted positive examples that are covered by a rule
$\overline{HB}$	Number of correctly predicted negative examples that are not covered by a rule
$H\bar{B}$	Number of incorrectly predicted negative examples that are not covered by a rule
$\hat{y}_i^j$	Prediction made by model j for example i
$\bar{M}$	Consensus model (or merged model)
N	Number of datasets/models
K	Total number of folds
$\mathcal{Z}$	Set of model evaluation specifications
$ \mathcal{Z} $	Number of evaluation specifications
Φ	Set of model evaluation metrics
Φ_r	Set of rule evaluation metrics
ϕ	Evaluation metric
$\mathcal{F}_C$	Set of conversion methods
$\mathcal{F}_M$	Set of merging methods
$\mathcal{S}$	Strata of operational models
Z_S	Set of evaluation results of operational models organized by strata
$\bar{\mathcal{M}}$	Set of consensus models
$Z_{\bar{\mathcal{M}}}$	Set of evaluation results of consensus models organized by strata
G	Number of metrics in a model evaluation specification
Z	Set of results of evaluation of operational model
$\bar{Z}$	Set of results of evaluation of consensus model
z_g	Result of evaluation using metric g
$I[p]$	Indicator function over a proposition p
I_x^M	Importance of variable x in model M

SYMBOLS

S	Stratum of operational models
ε	Stratification variable
$\mathcal{V}$	Set of stratification variables
J	Number of strata
L	Number of models in a stratum
$\mathcal{M}$	Set of models
$\mathcal{T}$	Set of test sets
T	Decision table
$\bar{T}$	Decision table resulting from merging a set of decision tables
h	Hyperrectangle
η	Result of decision tables merge attempt
α	Conflicting pair
β	Joining set
σ	Status of a merged rule
$\overline{\mathcal{R}}$	Merged rule-set
$\Delta\phi$	Performance loss of a consensus model relative to metric ϕ
$\overline{\Delta\phi}$	Average performance loss of a consensus model relative to metric ϕ
τ	Coverage threshold
$\hat{\text{dom}} x$	Empirical domain of variable x
$\mathcal{P}$	Power set
E	Synthetic rule dataset
t^{gen}	Multiset of generated values from subset t
C	Set of categories of a categorical variable
f	Relative frequency
x'	Symbolic variable
t'	Symbolic value
b	Bin
$\mathcal{B}$	Set of bins
$\mathcal{C}$	Category subset of a categorical variable empirical domain
q	Question
Q	Total number of questions
$p_{k,m}$	Probability of satisfying question m in rule k
κ	Cohen's Kappa
P_o	Observed agreement
P_e	Expected agreement
O_j	Set with all observed values of variable x_j
$\hat{p}$	Density estimate
Γ_f	Multivariate histogram
$\Gamma_{\hat{p}}$	Multivariate density estimator
w	Width of an edge of a hyperrectangle

SYMBOLS

V	Hypervolume of a hyperrectangle
d_c	Number of categorical variables
n_{mv}	Size of missing value
r_{mv}	Ratio of missing value
$\mathcal{X}_A$	Attribute space
$\mathcal{X}_D$	Instance space
$\mathcal{X}_M$	Learned space
$\mathcal{X}_h$	Space enclosed by hyperrectangle h

Chapter 1

Introduction

Data mining, an interdisciplinary field that merges computer science, statistics, and artificial intelligence, has been recognized for decades as a powerful tool to extract knowledge from data repositories (Han et al., 2012). Data mining techniques can be broadly categorized as descriptive or predictive. Descriptive data mining aims to uncover the underlying patterns and relationships in data, while predictive data mining focuses on forecasting future outcomes based on historical data. In both cases, the objective is to extract knowledge from data independently with varying degrees of human intervention.

For effective decision-making, a comprehensive understanding of the phenomenon under analysis is critical. This understanding often extends beyond mere predictions to encompass potential implications for strategic planning. Therefore, while it is essential to develop tools that accurately predict future events, it is also important to provide the rationale for the factors driving these outcomes. In the context of predictive models, the ability to understand these driving factors is referred to as *interpretability*. It is necessary to ensure accurate and interpretable models to satisfy both requirements. However, it is usually hard to establish an acceptable trade-off between the two characteristics, as models with high levels of accuracy tend to be less interpretable.

The problem of extraction of knowledge from models becomes even harder in the case of distributed models. In the era of globalized business models, many organizations now distribute their business processes across multiple locations. This dispersed nature of business processes presents a significant challenge for organizations aiming to extract useful knowledge from the vast amount of data generated across diverse locations and levels of detail. While individual models, particularly those based on rules like decision trees, are often designed for interpretability, this advantage offers limited usefulness when dealing with a multitude of models across a dispersed organizational structure. Specifically, the potentially large number of these models, each developed and applied in isolation, creates a significant barrier to understanding the phenomenon they collectively describe, even if each model is individually interpretable.

Hence, while locally developed tools are advantageous for capturing site-specific knowledge, their fragmented nature may hinder the organization's overall understanding of its processes and its ability to identify overarching patterns and trends. To illustrate the problem presented by dis-

tributed business processes, consider a multinational corporation with multiple retail outlets nationwide. In this scenario, each store may have its model to predict monthly sales performance. Similarly, a university with diverse academic programs can develop individual models for each course to evaluate student performance. A *business unit* is the smallest operational unit where the business is carried out and cannot be further decomposed. It also corresponds to the lowest level of decision-making. While analyzing processes within individual business units (e.g., retail stores, academic courses) aligns with local responsibilities, focusing exclusively on their models fails to provide a holistic view of the phenomena organization-wide. In the example of a multinational corporation, a retail company might overlook regional sales trends or correlations between store performance and customer demographics due to the fragmented nature of its modeling approach, which does not reveal the corporation's overall sales trajectory. Similarly, in the university example, the institution might fail to discern broader trends in student behavior or performance across different academic courses due to the partitioned nature of its modeling strategy, i.e., miss overarching trends in student retention or dropout.

This thesis focuses mainly on bringing interpretability to the knowledge dispersed across multiple related models. A novel approach is presented to iteratively merge multiple models into a single, interpretable model. The research began with a specific educational case study (Strecht et al., 2014a,b), where the methodology was initially developed and applied. It was then refined and generalized into a more flexible framework (Strecht et al., 2018, 2021) applicable to a broader range of problems. This framework, named *Inmplode*, provides a structured approach involving the creation and evaluation of individual operational models, grouping them into distinct strata based on shared characteristics, merging all models within each group using a heuristic, and evaluating the merged models using several metrics and criteria. The development process naturally raises several issues. Each one sparks distinct research directions, discussed in the remainder of this chapter.

1.1 Multiple models for related tasks: the case of predicting academic success

This research had its genesis in academic data analysis at the University of Porto, Portugal. Initially, the objective was to predict student performance, specifically, the likelihood of passing or failing a particular course at enrollment. To this end, decision trees were trained with socio-demographic data and past academic records. This case study exemplifies the limitations of localized models, motivating the need for a more holistic approach to understanding student success.

1.1.1 Decision levels in hierarchical organizations

Higher education institutions, with their inherently hierarchical structure and distinct organizational units, are a suitable starting point for exploring these problems. Universities or polytechnic

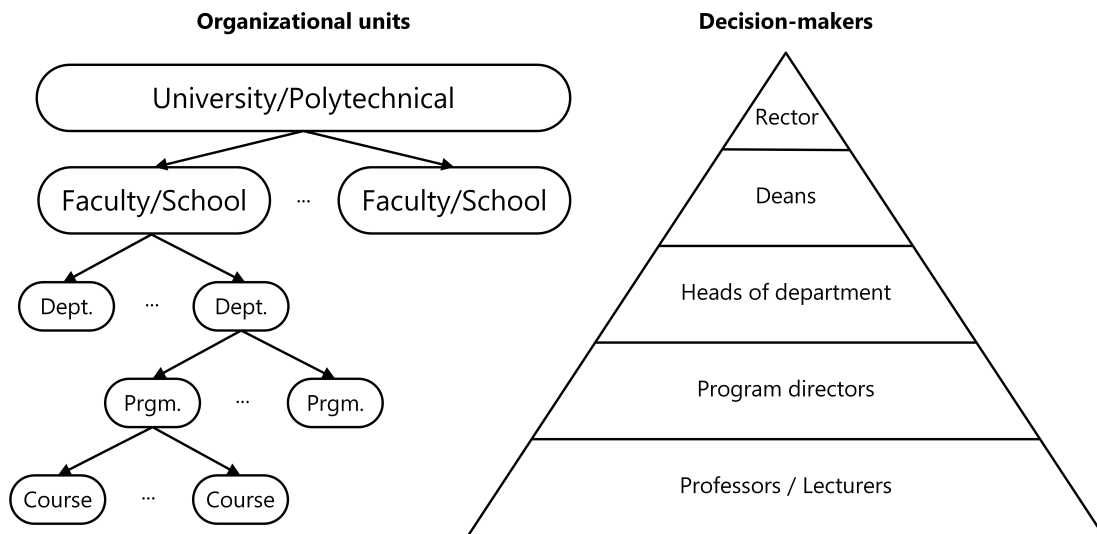


Figure 1.1: Decision-makers across the organizational units of a higher education institution

schools typically comprise faculties or schools subdivided into departments based on specific scientific areas. These departments deliver structured courses and programs in which students enroll. Completing all courses within a program is a requirement for graduation.

Universities, as complex hierarchical organizations, involve numerous decision-makers at different levels, as illustrated in Figure 1.1, which maps these roles to their corresponding organizational units. A typical hierarchy includes the rector or president at the top, responsible for the institution's overall strategic direction. Deans lead individual faculties or schools, managing resources and academic quality. Within faculties, heads of department handle daily operations, while Program directors oversee the syllabus of specific programs. At the course level, professors are responsible for teaching, research, and student supervision, often with the support of lecturers who focus on classroom instruction.

1.1.2 Separate models for related problems

Decision trees, a type of predictive and descriptive model, enable the prediction of individual student pass/fail outcomes as well as providing insights into the relationships between student attributes (e.g., attendance, prior grades) and those outcomes. This informs professors about student performance. Comparing these models across courses reveals how attribute influence varies. Consequently, a separate decision tree model was created for each course.

Hopefully, each model provides information to professors and lecturers to foster an increase in the number of students passing the course in future editions. Still, the knowledge contained within these models can also benefit other decision-makers. However, it is not feasible for a decision-maker such as a dean to meticulously examine all the models associated with the courses within their faculty. As the hierarchical decision level ascends, so does the number of models. The vast number of models and their narrow scope prevent a global view of the phenomenon under analysis. Furthermore, higher-level decision-makers are not primarily focused on understanding

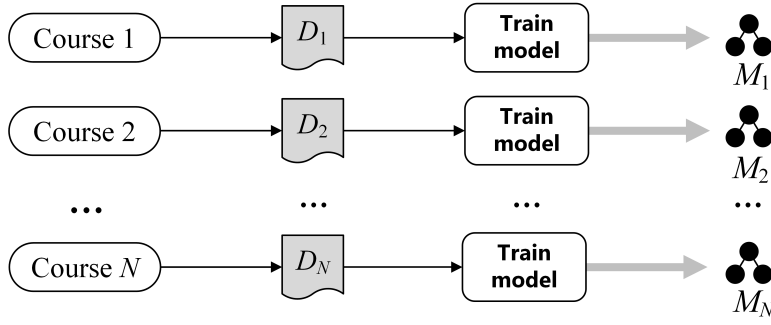


Figure 1.2: Creating professor/lecturer level models

behaviors restricted to individual models. Their objective is to discern patterns that extend across multiple models. This process of synthesizing insights from individual models to create a broader understanding of student success factors across the institution represents a form of *knowledge fusion*. In other words, for these higher-level decision-makers, the primary concern is to clarify the relationships between student attributes and the outcome of success or failure, irrespective of the specific course.

Considering the university case study, a viable approach to address the issue of having models targeted to each decision-maker could be establishing models based on the associated group of students. Accordingly, each model would be constructed exclusively from data from students enrolled in their respective courses, as shown in Figure 1.2. Each has its data from which the respective prediction models are created. At the program director level, a model would use data encompassing all students enrolled in program courses. Similarly, at the head of the department level, a model would employ data about students enrolled in programs administered by the department. At the subsequent level, a model for a dean would use data from the student population of all programs within a faculty/school. At the top-most level of decision making, a model for a rector would incorporate data from the entire student body of the higher education institution.

Figure 1.3 depicts the creation of a model (intended for analysis by decision-makers beyond the level of a professor/lecturer) using a set-up akin to that of Figure 1.2. However, the dataset is the union of datasets from all courses relevant to the specified decision level. Although intuitively appealing, directly concatenating datasets and creating a single model per decision level

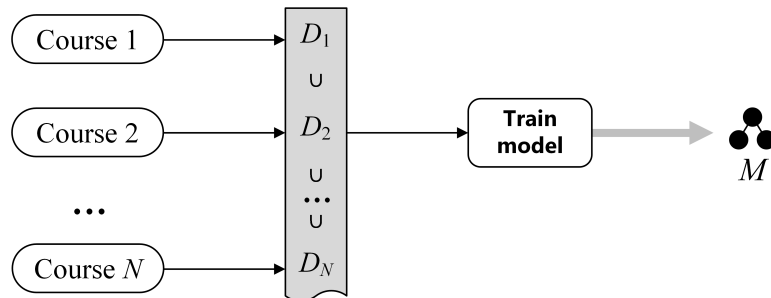


Figure 1.3: Creating a single decision level model by dataset concatenation

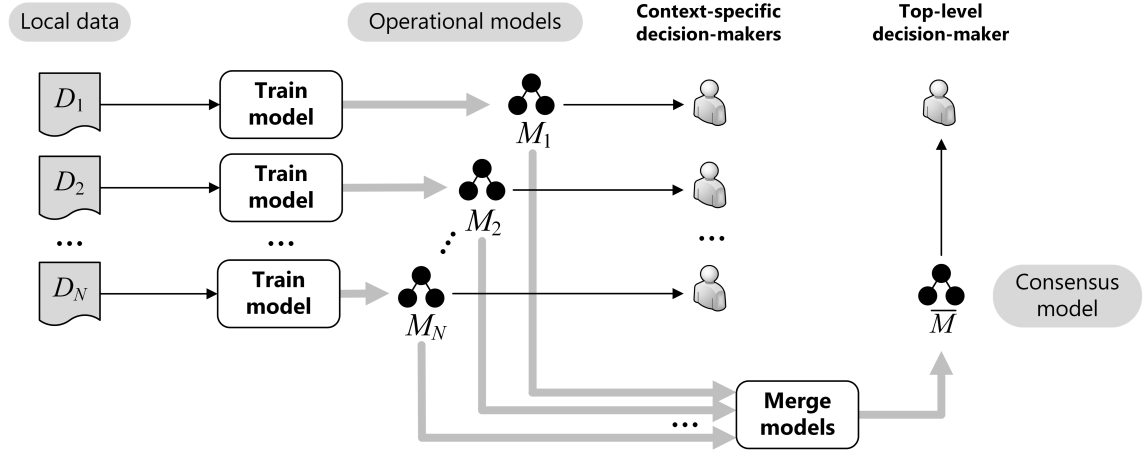


Figure 1.4: Merging operational models into a consensus model

oversimplifies the complexities of student performance across diverse courses. This single-model approach treats students within a decision level as homogeneous, disregarding unique assessment methods, learning objectives, and pedagogical styles in each course. Consequently, this homogenization leads to an overgeneralized model that fails to capture the subtlety of student behavior across diverse courses.

1.2 Consensus models as a holistic representation of knowledge

In distributed systems, individual operational models are frequently developed to predict outcomes within specific sub-domains. An *operational model* is used within a specific context or business unit, focusing on predicting or explaining phenomena within a limited scope. These models are narrow in scope, limited to the specific data and characteristics of their respective sub-domains. However, higher-level decision-making within these systems requires a more holistic perspective. The challenge lies in achieving a representation of knowledge that goes beyond the scope of these business units. For example, for strategic planning and resource allocation, understanding aggregated behavioral patterns across multiple sub-domains becomes critical. Directly using raw data from different sub-domains can obscure important contextual variations, potentially hindering granular analysis. Furthermore, higher-level decisions typically operate at a coarser level of granularity. Therefore, managing and individually analyzing numerous domain-specific operational models is inefficient for decision-makers requiring concise, aggregated information relevant to system-level strategies.

To address these limitations, this work proposes the approach of merging multiple existing operational models to create a *consensus model*. Rather than preserving all operational model subtleties, the consensus model identifies shared patterns and trends across contexts, offering a more generalizable and interpretable representation. This approach, illustrated in Figure 1.4, presents a method for creating consensus models based in merging operational models.

The goal of merging operational models into consensus models is to achieve both a holistic interpretation and generalized prediction. The aim is to overcome the limitations of individual operational models by providing a unified description of system behavior across the entire organization, eliminating the need for separate analysis of each model. Additionally, consensus models seek to offer reasonable predictions across diverse contexts, capturing overarching trends while acknowledging that there may be a reduction in predictive performance compared to highly specialized, context-specific models.

In the case study, this tiered approach provides targeted information to different stakeholder levels while preserving the usefulness of operational models. Program directors use consensus models derived from the merging of course models specific to their programs for program-level decisions. Department heads rely on models that include related programs for broader strategic planning. Deans use faculty-wide consensus models for a comprehensive understanding at the institutional level. Rectors access models that encompass the entire institution, providing a holistic view. This hierarchy shows that higher decision-making levels require fewer and more aggregated models, aligning knowledge with the relevant decision level, without compromising the detailed information available through the operational models.

1.3 Hypothesis and approach

This thesis investigates the challenge of consolidating knowledge from multiple, interrelated, rule-based models into a smaller set of consensus models. While the research is initially motivated by a specific case study in the University of Porto, the underlying questions and implications extend beyond this particular scope. This study delves into the fundamental principles and inherent limitations of knowledge merging within rule-based systems. To achieve this, the research explores these challenges through empirical analysis across multiple datasets, aiming to generalize findings beyond the initial case study. To address these challenges, this research proposes and tests the following hypothesis, which articulates the core premise of the study.

Hypothesis: Machine learning methods can be used to generate a set of consensus models that provide a concise and representative summary of the knowledge contained within multiple rule-based models. The representativeness of these consensus models can be evaluated using metrics that assess predictive performance, fidelity to the original models, interpretability, and completeness of the decision space.

This research began with the use of decision tree models to predict student academic performance at the University of Porto. Initially, the focus was on a limited set of courses, but as the study progressed, it expanded along multiple dimensions. These included increasing the number

of models merged, exploring different grouping criteria for merging, and contextualizing the predictive performance of decision trees relative to other models. Additionally, the impact of merging order on model behavior was investigated. These initial explorations underscored the need for a more systematic and structured approach to model merging, motivating the development of a dedicated framework.

The literature presents a variety of approaches to the many tasks involved in model merging, each with different assumptions and constraints. To address this diversity and provide a unified methodology, this research formalizes the process within a comprehensive and adaptable framework named *Inmplode*. This framework guides the training and evaluation of operational models, their stratification into groups, and the development and evaluation of consensus models. Furthermore, it incorporates placeholders for specialized components that handle key tasks in model merging. These tasks involve selecting appropriate merging methodologies, defining the sequence in which models are combined, resolving prediction conflicts arising from overlapping rules, and simplifying the final rule set to improve interpretability. The framework was also designed for generalizability, accommodating various rule-based models and datasets. A key feature is the conversion of operational models into decision tables, which standardizes the representation to streamline the merging process.

To assess the effectiveness of the consensus models generated by *Inmplode*, an empirical evaluation was performed across five datasets. The assessment considered not only predictive performance, but also fidelity - defined as the agreement between the consensus model and the original operational models - as well as interpretability and completeness. The results provided several key insights, one of which was that, in some cases, the consensus models exhibited a degree of incompleteness, a consequence of the merging process itself. This limitation stems from the fact that consensus models, while effectively summarizing knowledge, often fail to provide predictions for all possible combinations of independent variables within the decision space.

Given that incompleteness is a significant challenge and can undermine the purpose of consensus models, the research shifted focus to methods to improve completeness. Traditional model creation relies on datasets with single-valued observations; however, the incomplete rules generated during the merging process require alternative approaches. The first method involved generating synthetic data for use with standard propositional decision tree algorithms. The second applied the Symbolic Data Analysis paradigm, converting rules into distributional data that capture the range and distribution of values within each rule. This led to the development of distributional data trees, using a specialized algorithm designed for this type of data, which improved both the completeness and the interpretability of consensus models.

The evaluation also revealed a decrease in predictive performance and fidelity of the consensus models compared to the operational models, limiting their ability to act as direct surrogates for the latter. This highlights their potential as explanatory tools for knowledge consolidation, rather than as replacements for operational models in prediction tasks. Furthermore, research included a correlation analysis between different groups of evaluation metrics across the five datasets. This analysis focused on aspects such as predictive performance, fidelity, interpretability, and completeness.

The identified correlations reveal the inherent trade-offs in model merging, providing practitioners with valuable insights for effectively deploying the *Inmplode* framework in real-world scenarios.

1.4 Contributions and publications

1.4.1 Main contributions

This thesis presents a number of contributions to the field of model merging, each of which is directly related to the hypothesis, described next.

1.4.1.1 The *Inmplode* framework for merging rule-based models

This research introduces *Inmplode*, a novel framework for generating consensus models by merging multiple rule-based models. The framework addresses the challenges outlined in the hypothesis, focusing on the creation of concise and representative models that encapsulate the knowledge embedded within the operational models. Chapters 4, and 5 details the development and evaluation of *Inmplode* also presented in the following publications:

- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2014). “Merging Decision Trees: A Case Study in Predicting Student Performance”. In: Luo, X., Yu, J.X., Li, Z. (eds) Advanced Data Mining and Applications. ADMA 2014. Lecture Notes in Computer Science, vol 8933. Springer, Cham ([Strecht et al., 2014b](#)) (**Best Paper Award**)
- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2021), “Inmplode: A Framework to Interpret Multiple Related Rule-based Models”. Expert Systems. 2021; 38:e12702 ([Strecht et al., 2021](#))

Additional aspects of this contribution, including preliminary results and specific applications of the framework, have been published in the following:

- Pedro Strecht, Luís Cruz, Carlos Soares, Joao Mendes-Moreira, Rui Abreu (2015). “A Comparative Study of Classification and Regression Algorithms for Modelling Students’ Academic Performance”, 8th International Conference of Educational Data Mining (EDM2015) ([Strecht et al., 2015a](#))
- Pedro Strecht (2015). “A Survey of Merging Decision Trees Data Mining Approaches”. In A. A. Sousa and E. Oliveira, editors, 10th Doctoral Symposium in Informatics Engineering (DSIE’15) ([Strecht, 2015](#))
- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2015). “Exploring the Impact of Ordering Models in Merging Decision Trees: A Case Study in Education”, Doctoral Consortium, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. ECML PKDD 2015 ([Strecht et al., 2015b](#))

- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2018). “A Framework for Analytical Approaches to Combine Interpretable Models”. In: Lossio-Ventura, J., Muñante, D., Alatrasta-Salas, H. (eds) Information Management and Big Data. SIMBig 2018. Communications in Computer and Information Science, vol 898. Springer, Cham (Strecht et al., 2018)
- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2019). “Generalizing Knowledge in Decentralized Rule-Based Models”. In: Monreale, A., et al. ECML PKDD 2018 Workshops. ECML PKDD 2018. Communications in Computer and Information Science, vol 967. Springer, Cham (Strecht et al., 2019)

1.4.1.2 Methods to complete consensus rule-based models

This research introduces two novel approaches to address the challenge of incomplete consensus models. The first, Synthetic data trees, generates synthetic data to represent decision space regions defined by rules, which is then used to build a model using standard propositional decision tree algorithms. The second, Distributional data trees, represents the rule conditions as probability distributions, constructing a decision tree using a specific algorithm tailored for distributional data. The detailed development and empirical evaluation of these approaches are presented in Chapter 6 and published in:

- Pedro Strecht, João Mendes-Moreira, Carlos Soares (2024). “Symbolic Data Analysis to Improve Completeness of Model Combination Methods”. In: Liu, T., Webb, G., Yue, L., Wang, D. (eds) AI 2023: Advances in Artificial Intelligence. AI 2023. Lecture Notes in Computer Science, vol 14472. Springer, Singapore (Strecht et al., 2024)

1.4.2 Additional contributions

This research also presents contributions related to evaluating the completeness of consensus models, a key aspect of assessing their representativeness as outlined in the hypothesis. These contributions arise from the need for new evaluation measures in the context of model merging. While supportive rather than core elements, they are important for understanding and quantifying completeness. Due to space constraints, these contributions are detailed in the appendices.

1.4.2.1 Rule-based models completeness metrics

To address the limitations in evaluating the completeness of consensus models, this research introduces two novel metrics: geometrical completeness and distributional completeness. These metrics provide quantitative measures of the model’s coverage of the decision space. The metrics were validated across five datasets, demonstrating their effectiveness in assessing completeness. The details of their development and evaluation are presented in the following publication:

- Pedro Strehct, João Mendes-Moreira, Carlos Soares (2025). “Estimating Completeness of Consensus Models: Geometrical and Distributional Approaches”. In: Nicosia G., Pardalos, P., Ojha V. (eds) The 10th International Conference on Machine Learning, Optimization and Data Science. LOD 2024. Springer, Cham (Strehct et al., 2025).

1.4.2.2 Density estimation method for high-dimensional data

To support the calculation of distributional completeness, a novel density estimation method based on multivariate histograms was developed. Existing methods are unable to adequately address the challenges in measuring distributional completeness, particularly for high-dimensional datasets with categorical variables and missing data. This new method overcomes these limitations, offering a more effective solution. The method was evaluated across several datasets, with its development and results published in:

- Pedro Strehct, João Mendes-Moreira, Carlos Soares (2022). “Density Estimation in High-Dimensional Spaces: A Multivariate Histogram Approach”. In: Chen, W., Yao, L., Cai, T., Pan, S., Shen, T., Li, X. (eds) Advanced Data Mining and Applications. ADMA 2022. Lecture Notes in Computer Science, vol 13726. Springer, Cham (Strehct et al., 2022)

1.5 Thesis outline

The remainder of this thesis is organized into chapters and appendixes as follows:

- Chapter 2 provides the foundational concepts for this research, introducing key machine learning principles and exploring rule-based models. It covers their various forms and representations, which are integral to the methodologies developed and applied throughout the study.
- Chapter 3 frames the research within the existing literature, introducing concepts and terminology related to model interpretability and merging. It provides an overview of interpretability techniques and evaluation metrics, compares model merging with ensemble learning, and explores the analytical and mathematical approaches to merging prediction models.
- Chapter 4 presents *Inmplode*, a novel framework designed to generate consensus models from multiple, context-specific rule-based operational models. It outlines the core problem the framework addresses, describes its modular architecture, and highlights the distinction between its general workflow and customizable components.
- Chapter 5 offers a comprehensive analysis of the model merging process, a core aspect of *Inmplode*. It details the methods used to combine multiple rule-based models into a single rule-based model, supported by an empirical evaluation based on a case study from the University of Porto.

- Chapter 6 addresses the issue of incompleteness in consensus models. Two approaches for constructing decision trees from incomplete rule sets are presented: the first generates a synthetic dataset from the rules and applies conventional decision tree methods, while the second converts the rules into distributional data, using a specialized algorithm for this type of data.
- Chapter 7 presents the results of the empirical evaluation of the *Inmplode* framework across five datasets. It covers three model representations (merged tables, synthetic data trees, and distributional data trees). The evaluation focuses on predictive performance, fidelity, interpretability, and completeness, highlighting trade-offs for practical applications.
- Chapter 8 summarizes the contributions of this thesis, particularly in relation to the hypothesis of using machine learning methods to generate representative consensus models. It also identifies key areas for future research, highlighting opportunities to further refine the process of merging rule-based models and improving their representativeness.
- Appendix A presents a method for estimating dataset density using a multivariate histogram, aimed at improving the measurement of completeness in consensus models within *Inmplode*. It features the method's effectiveness in handling high-dimensional datasets, missing data, and dataset variations, validated through empirical evaluation across several datasets.
- Appendix B introduces two novel metrics for evaluating the completeness of rule-based models within the *Inmplode* framework. The first metric evaluates model completeness evenly across the decision space, while the second evaluates completeness relative to data concentration using the density estimation technique from Appendix A. Both methods are evaluated on selected datasets.
- Appendix C details the datasets used in experiments for creating operational and consensus models with *Inmplode*. It also describes the business units for model creation, stratification criteria, and consensus models, further expanding on the empirical evaluation results presented in Chapter 7.

Chapter 2

Background

This chapter lays the groundwork for the research presented in this document by providing an overview of fundamental concepts and establishing consistent terminology and notation. The chapter is organized in two sections. The first section introduces key machine learning concepts relevant to the research questions explored in subsequent chapters. The second section discusses the specific domain of rule-based models, examining their various forms and representations, which are central to the methodologies employed in this study.

2.1 Machine learning essential concepts and fundamentals

2.1.1 Knowledge discovery, data mining, and machine learning

Within the landscape of data analysis, three distinct yet interconnected concepts emerge. The first is *Knowledge Discovery in Databases* (KDD), defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad et al., 1996). Other definitions refer to extracting implicit, previously unknown, and likely valuable information from data, typically devising models and patterns from large databases (Sarker et al., 2000; Bramer, 2007). The second is *Data Mining* (DM), which, while often used as a synonym for KDD, represents a more specific step within the broader KDD process (Fayyad et al., 1996). Data mining merges conventional data analysis methods with algorithms that, within acceptable computational efficiency constraints, identify patterns over vast amounts of data (Tan et al., 2006). The third is *Machine Learning* (ML), a field of computer science that enables computers to learn from data without explicit programming, uncovering knowledge that would be challenging or impossible to deduce through traditional statistical analysis or manual human data inspection (Mitchell, 1997). While considered a subfield of DM, its emphasis on automation is the differentiator. Traditional data mining does not necessarily require this automated learning aspect. Thus, it refers to how computer systems use computer science and statistical techniques to progressively improve their performance on analytical tasks by learning from data (Samuel, 1959).

The terms *data mining* and *machine learning* are often used interchangeably in the literature. Due to classical statistics being the cornerstone of both DM and ML, it is not entirely feasible to

separate these two domains. They have mutually influenced, inspired, learned from, and adopted each other throughout their development and evolution (Shu and Ye, 2023).

When the term was introduced in the late 1950s (Samuel, 1959), ML represented a transformative shift in data analysis, offering the ability to construct models from algorithms that iteratively learn and refine themselves based on data (Shu and Ye, 2023). This approach is often described as *training* computers to identify patterns within data (Fayyad et al., 1996), which can be used for predictive modeling, pattern recognition, and ultimately, informed decision-making. ML presents a significant change for practitioners with a background in traditional software development methodologies, where explicit instructions are central, as it relies on an indirect programming approach where data, rather than detailed instructions, guides the learning process (Molnar, 2019). This enables computers to discover patterns autonomously with minimal human intervention.

ML is commonly divided into two major areas. The first, *supervised learning*, encompasses a suite of methods that, by analyzing data, can make predictions about new data related to a specific phenomenon. Supervised learning encompasses tasks such as classification and regression, where an algorithm is provided with explicit input-output pairs, guided by a human supervisor, aiming to discern a mapping relationship expressed through a model (Bishop, 2006; Hastie et al., 2017). The second central area within ML is *unsupervised learning*, which seeks to reveal latent relationships and patterns within the data that otherwise might not be immediately apparent (Murphy, 2012; Hastie et al., 2017). Both areas encompass a wide range of algorithms and techniques.

As illustrated in Figure 2.1 (Fayyad et al., 1996; Krzysztof J. Cios and Witold Pedrycz and Roman W. Swiniarski, 2007), KDD is a multi-stage process for extracting knowledge from raw data. Due to the inherent heterogeneity of data sources, an initial stage of data integration and selection is often necessary. This stage involves consolidating data from disparate locations, databases, and systems into a designated repository. Following selection, the chosen dataset undergoes preprocessing, an important stage designed to enhance its suitability for automated analysis. Preprocessing encompasses various tasks, including noise removal, handling missing data, and defining strategies for managing incomplete information.

The *preprocessed data* then enters the essential stage of pattern discovery. This stage aims to reveal latent patterns and relationships within the data. It uses a diverse toolkit encompassing machine learning algorithms or various data mining techniques, including manual analysis when appropriate. These techniques unveil the hidden structure and relationships within the data, represented as models or rules. However, this raw depiction of knowledge requires further scrutiny.

The subsequent interpretation and evaluation stage holds paramount importance. This stage meticulously assesses the validity and significance of the extracted knowledge. Rigorous evaluation techniques ensure the patterns accurately reflect the underlying data and align with the research objectives. This stage may necessitate revisiting previous stages for further data preparation, algorithm selection, or refinement of the analysis techniques. This iterative approach ensures the extracted knowledge is meaningful but also reliable and generalizable.

Finally, the KDD process culminates in the consolidation of discovered knowledge. This involves integrating the knowledge into other systems for automated decision-making, visualizing

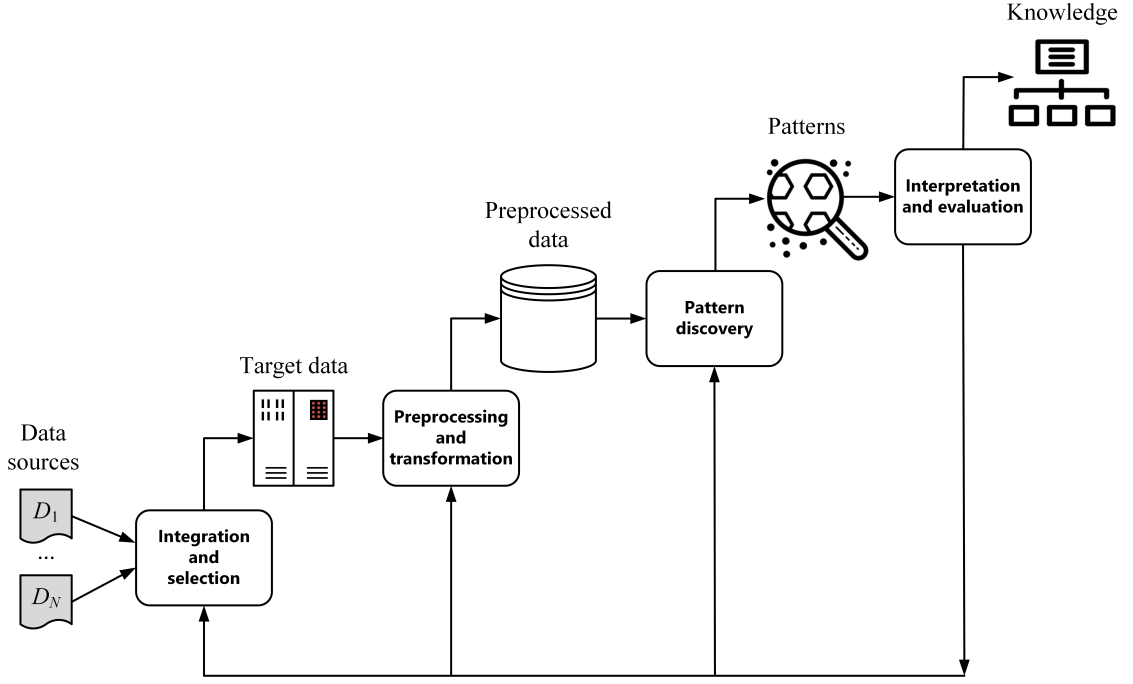


Figure 2.1: The Knowledge Discovery in Databases process

the patterns for enhanced comprehension, or documenting it for future reference.

2.1.2 Data representation and related concepts

2.1.2.1 Examples and variables

In machine learning, data is typically represented as a collection of examples. Each *example*, formally denoted as e_i , represents a single real-world entity and is structured as a tuple containing specific values assigned to relevant variables (Equation 2.1). A variable, denoted as x , is a data unit, capturing specific characteristics or measurements associated with an entity (Han et al., 2012). Therefore, d represents the total number of variables used to describe the example, and each $v_{i,j}$ denotes the specific value observed for variable x_j within a particular example e_i .

$$e_i = (v_{i,1}, v_{i,2}, \dots, v_{i,d}) \quad (2.1)$$

Variables are broadly classified into two categories: categorical and numerical. *Categorical variables* represent distinct categories and are further divided into *nominal* when categories do not have an intrinsic order (e.g., colors, marital status) and *ordinal* when categories have a defined order (e.g., educational levels, movie ratings). On the other hand, *numerical variables* hold numerical values and are used for quantitative measurements. They are further categorized as *discrete* when the values are distinct and countable (e.g., number of items in a set, word count in a document) or *continuous* when they can assume any value within a continuous range (e.g., temperature, weight, time).

Some terms are often used interchangeably in machine learning. The term *example* encompasses numerous alternative terms, including *instance*, *observation*, *object*, *tuple*, *data point*, and *statistical unit*. Similarly, the term *variable* is often used as a synonym of *attribute*, *dimension*, or *feature*. While some authors may draw subtle distinctions between these terms in specific contexts, this thesis predominantly employs *example* and *variable* in singular and plural forms. Some alternative terms may be used in specific cases, or distinctions between them may be established when necessary.

2.1.2.2 Domain of a variable

The concept of *domain* is essential for understanding the characteristics and potential variations within a dataset. Each variable has a well-defined domain, denoted by $\text{dom } x$, encompassing all possible values it can take on. This domain can be categorized into two types, depending on the type of the variable:

- Discrete domains – For both categorical variables (representing distinct categories) and discrete numerical variables (assuming finite, whole number values), the domain is represented as a finite set of values (Equation 2.2). Each distinct value the variable can take is denoted by c_j , with m representing the total number of possible values within the set. This implies a well-defined collection of categories or numbers for the variable.
- Continuous domains – For continuous numerical variables, which can assume any value within a specific range, the domain is represented by an interval (Equation 2.3). This interval reflects that these variables can take on unlimited values within a continuous range. The domain's lower and upper bounds are denoted by l and u , respectively.

$$\text{dom } x = \{c_1, c_2, \dots, c_m\} \quad (2.2)$$

$$\text{dom } x = [l; u] \quad (2.3)$$

Examples of variable domains include:

- $\text{dom } x = \{\text{single, married, widowed, other}\}$, a set for a categorical nominal variable;
- $\text{dom } x = \{\text{primary, secondary, tertiary}\}$, a set for a categorical ordinal variable;
- $\text{dom } x = \{1, 2, 3, 4, 5\}$, a set for a discrete numerical variable;
- $\text{dom } x = [18; 100]$, an interval for a continuous numerical variable.

2.1.2.3 Datasets

Another pivotal concept in machine learning data representation is the dataset. Formally denoted as D , a *dataset* is an unordered collection of examples (Equation 2.4). Each example represents a single entity from a real-world phenomenon, like a customer in a sales dataset or a student in a university enrollments dataset. The *cardinality* (or *size*) of a dataset refers to the number of examples it contains, denoted by $n = |D|$.

$$D = \{e_1, e_2, \dots, e_n\} \quad (2.4)$$

Background

The variables associated with each example within a dataset D can be grouped into a set denoted as X (Equation 2.5). The *dimensionality* of a dataset refers to the number of variables it encompasses, denoted as $d = |X|$. This concept is analogous to the dimensionality of geometric objects: the number of dimensions represents the number of independent axes or directions along which a point can vary. For example, a point in 3-D space is uniquely defined by its coordinates on three axes (x_1, x_2, x_3) . Similarly, each example in a dataset is represented by a set of values corresponding to its variables, forming a multi-dimensional space. Consequently, adding more variables to a dataset increases its dimensionality.

$$X = \{x_1, x_2, \dots, x_d\} \quad (2.5)$$

Datasets are often represented in a tabular format, resembling a spreadsheet (Table 2.1). Each column represents a variable, while each row represents an individual example (Equation 2.1).

The *volume* of a dataset, denoted as μ , corresponds to the product of the dataset's size (n) and its dimensionality (d), capturing the overall amount of data within the dataset (Equation 2.6).

$$\mu = n \times d \quad (2.6)$$

Table 2.1: Structure of a dataset

	x_1	x_2	$\dots$	x_d
e_1	$v_{1,1}$	$v_{1,2}$	$\dots$	$v_{1,d}$
e_2	$v_{2,1}$	$v_{2,2}$	$\dots$	$v_{2,d}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
e_n	$v_{n,1}$	$v_{n,2}$	$\dots$	$v_{n,d}$

2.1.3 Training of prediction models

Classification and regression are predictive modeling techniques with distinct applications in several domains, from data science and machine learning to business intelligence and marketing analytics. The essence of their distinction lies in the output type they predict: classification models assign examples to predefined categories (or classes), while regression models estimate continuous numerical values.

Classification addresses prediction problems by assigning examples to predefined *classes*, corresponding to categories. For instance, a classification model could predict whether a student will pass or fail a course or if a customer churns (cancels a service). Also known as *classifiers*, these models learn from datasets where each example is associated with a known class. Through this process, the patterns and relationships between *independent variables* and the corresponding classes of a *target variable* are discovered and assembled in a *prediction model*. The model is

Table 2.2: Structure of a training set

	x_1	x_2	$\dots$	x_d	y
e_1	$v_{1,1}$	$v_{1,2}$	$\dots$	$v_{1,d}$	y_1
e_2	$v_{2,1}$	$v_{2,2}$	$\dots$	$v_{2,d}$	y_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
e_n	$v_{n,1}$	$v_{n,2}$	$\dots$	$v_{n,d}$	y_n

then used to give a *prediction* of the value of the target variable for any new example. Classification tasks vary in complexity (Sokolova and Lapalme, 2009). The simplest scenario is *binary classification*, where an example is assigned to one of two distinct classes. *Multi-class classification* involves assigning an example to one of several non-overlapping classes. The most complex case is *multi-labeled classification*, where an example can be assigned to multiple non-overlapping classes (Han et al., 2012).

Regression, on the other hand, focuses on predicting continuous-valued numbers. For example, a regression model could predict a student’s final exam mark or estimate a product’s sales over time. These models also referred to as *regressors*, extract patterns and relationships between the independent variables and the corresponding target values. Like classifiers, regressors can predict target values for new examples.

In either problem, datasets form the foundation for developing and evaluating predictive models. Typically, a dataset is partitioned into two subsets: a training set and a test set. A *training set*, denoted as Tr , is a dataset used to train a prediction model. Regarding the variables, the set of *independent variables*, denoted as X_{Tr} (Equation 2.7), is treated separately from the *target variable*, denoted as y , which the model predicts. The complete set of variables V_{Tr} encompasses both (Equation 2.8), with d , in this context, denoting the *dimensionality* of the training set, i.e. considering only the number of independent variables ($d = |X_{Tr}|$)

$$X_{Tr} = \{x_1, x_2, \dots, x_d\} \quad (2.7)$$

$$V_{Tr} = X_{Tr} \cup \{y\} \quad (2.8)$$

An example within a training set (Equation 2.9) is a tuple that includes values for both independent variables ($v_{i,j}$) and the target variable (y_i). Similar to the dataset in Table 2.1, a training set can be structured in a tabular format, with columns representing the variables and rows representing individual examples (Table 2.2). In this context, n denotes the size of the training set ($n = |Tr|$).

$$e_i = (v_{i,1}, v_{i,2}, \dots, v_{i,d}, y_i) \quad (2.9)$$

A *model*, denoted as M , is the outcome of analyzing a dataset to formulate a mapping function between the values of independent variables and the corresponding values of the target variable across all examples. A *learning algorithm* or learner is any procedure used to build a model (Gama, 1999), denoted as $\mathcal{L}$. Both classification and regression encompass a large set of algorithms with

diverse approaches and underlying principles, providing flexibility in selecting the most suitable method based on the specific characteristics of the dataset and the problem at hand.

Formally, the Train function (Equation 2.10) uses a learning algorithm ($\mathcal{L}$) and a training set (Tr), with independent variables (X_{Tr}) and a target variable (y), to create a model (M). The Predict function (Equation 2.11) then uses this model (M) to estimate the target variable for a given example (e), yielding a prediction denoted by $\hat{y}$.

$$M = \text{Train}(\mathcal{L}, Tr, X_{Tr}, y) \quad (2.10) \quad \hat{y} = \text{Predict}(M, e) \quad (2.11)$$

The term *predictor* has two meanings: it can refer to a prediction model (Breiman, 1996; Mendes-Moreira et al., 2012), or it can refer to an independent variable (also known as a predictor variable) within a model, where the target variable is the dependent or response variable (Green, 1978; Han et al., 2012; Mizumoto, 2023).

2.1.4 Evaluation of predictions models performance

Evaluating the performance of predictive models is crucial for selecting the most suitable approach for a given problem. This process involves making predictions on a test set, denoted as Ts , which shares the same structure as the training set but consists exclusively of examples not used during model training. This separation ensures an unbiased assessment of the model's ability to generalize to new, unseen data.

In classification problems, performance evaluation requires distinguishing between correct predictions and errors based on the agreement between the model's outputs and the true target values. The class of interest, often carrying greater significance due to the consequences of misclassification, is designated as the *positive class* and denoted as $\oplus$ (Han et al., 2012). *True positives (TP)* correspond to instances correctly classified as belonging to the positive class, while *true negatives (TN)* represent instances correctly classified as belonging to the negative class. *False negatives (FN)* occurs when instances belonging to the positive class are incorrectly classified as negative, whereas *false positives (FP)* refer to instances from the negative class that are mistakenly classified as positive. Thus, true positives and true negatives represent correct predictions, aligning with reality, while false positives and false negatives constitute classification errors.

These class-specific counts are often presented in a contingency table known as *confusion matrix* (Lavrač et al., 1999), shown in Table 2.3. This matrix summarizes the model's performance by capturing the agreement and disagreements between its predictions and the actual values.

Table 2.3: Confusion matrix

		Predicted values	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

2.1.4.1 Metrics to estimate a model's predictive performance

One of the most commonly used metrics for evaluating predictive performance is *Accuracy*, which measures the proportion of examples in a test set that the model classifies correctly (Equation 2.12). Ranging from 0 to 1, accuracy indicates the model's performance, with 1 representing perfect classification of all positive and negative examples and 0 representing complete misclassification of all examples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.12)$$

Accuracy, though widely used, can be misleading in imbalanced datasets, where one class of the target variable significantly outnumbers the others. In such cases, a model can achieve high accuracy by predicting only the majority class, neglecting the minority class entirely. To address this limitation, metrics such as precision and recall are preferred, as they focus specifically on the model's performance with respect to the positive class (frequently the minority class). Both precision and recall are relative metrics, ranging from 0 to 1. *Precision* (Equation 2.13) measures the proportion of correctly classified positive examples among all positive predictions, reflecting the reliability of the model's positive predictions. *Recall* (Equation 2.14), on the other hand, measures the proportion of correctly classified positive examples out of all actual positive cases, highlighting the model's ability to minimize false negatives and capture all relevant positive examples.

$$Precision = \frac{TP}{TP + FP} \quad (2.13) \quad \quad \quad Recall = \frac{TP}{TP + FN} \quad (2.14)$$

To provide a balanced evaluation of both precision and recall, the F1 score (Equation 2.15) is often used. This metric, which is the harmonic mean of precision and recall, combines them into a single value ranging from 0 to 1. A score of 1 indicates perfect precision and recall, while a score of 0 signifies that either precision or recall is zero. The F1 score is particularly useful for balancing the trade-off between precision and recall, especially in scenarios involving imbalanced datasets.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \quad (2.15)$$

Cohen's Kappa, denoted by κ (Equation 2.16), is a chance-corrected alternative to Accuracy, accounting for agreement that could occur by chance. The formula compares *observed agreement*, denoted as P_o , which corresponds to Accuracy, with *expected agreement*, denoted as P_e (Equation 2.17), considering the distribution of actual positives and negatives in the test set. Similarly, the *Matthews Correlation Coefficient* (MCC) (Equation 2.18) provides a measure of agreement that addresses the limitations of both Accuracy and the F1 score.

$$\kappa = \frac{P_o - P_e}{1 - P_e} \quad (2.16)$$

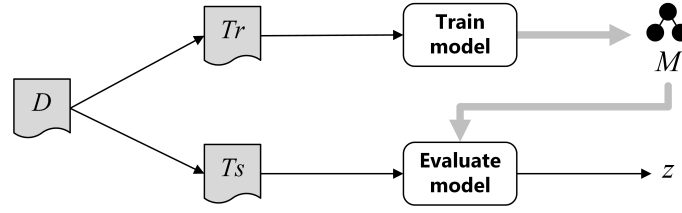


Figure 2.2: Estimation of performance with the holdout method

$$P_e = \frac{(TP + FP)(TP + FN) + (FP + TN)(FN + TN)}{(TP + TN + FP + FN)^2} \quad (2.17)$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.18)$$

2.1.4.2 Methods to estimate a model's performance

An important assumption in model evaluation is that the test set consists only of examples not used during training, ensuring an unbiased measure of the model's predictive performance. To partition data for training and testing, two common methods are employed: holdout and k -fold cross-validation. The selection of the most suitable evaluation method relies on several factors: dataset characteristics (size, class distribution, imbalanced classes), data availability (limited resources may influence method choice), and specific modeling requirements (complexity of the model and computational constraints) (Han et al., 2012).

The *holdout* method is a simple approach to estimating the predictive performance of models. In this method, a dataset D is partitioned into two distinct subsets, with the training set typically comprising two-thirds of the data and the test set encompassing the remaining one-third of the data (Han et al., 2012). Figure 2.2 presents a visual representation of the holdout method. The evaluation can use any metric resulting in a performance z .

Holdout, while simple, relies on a single training-test split, which can lead to biased performance estimates depending on the specific partitioning of the data. To mitigate this issue, *random subsampling* (also referred to as *Monte Carlo cross-validation*) (Picard and Cook, 1984) repeats the holdout method k times, each time randomly sampling new training and test sets from the original data. The model is trained and evaluated for each split, and the average performance is evaluated across iterations. However, random subsampling still allows data points to appear in multiple training sets while others may never be included in a test set, leading to potential biases in the evaluation.

An alternative that addresses this limitation is *k-fold cross-validation*, which provides a more systematic approach to partitioning the dataset. Unlike Monte Carlo cross-validation, where data splits vary across iterations, k -fold cross-validation ensures that each data point is used for both training and testing exactly once. In this method, the dataset is randomly partitioned into k disjoint

(non-overlapping) subsets of roughly equal size (Equation 2.19), known as *folds* (Stone, 1974). Each example belongs to a single, specific fold (Equation 2.20).

$$D = f_1 \cup \dots \cup f_k \quad (2.19) \quad f_i = \{e_1, \dots, e_{|f_i|}\} \quad (2.20)$$

The model undergoes training and evaluation in k iterations. During each iteration, one fold (f_i) is designated as the test set (Equation 2.22), while the remaining $k - 1$ folds are assembled to form the training set (Equation 2.21). This ensures that every example is used for both training and testing, with each example participating in the training process the same number of times and being used for testing exactly once. Due to its relatively low bias and variance, 10-fold cross-validation is generally recommended (Han et al., 2012).

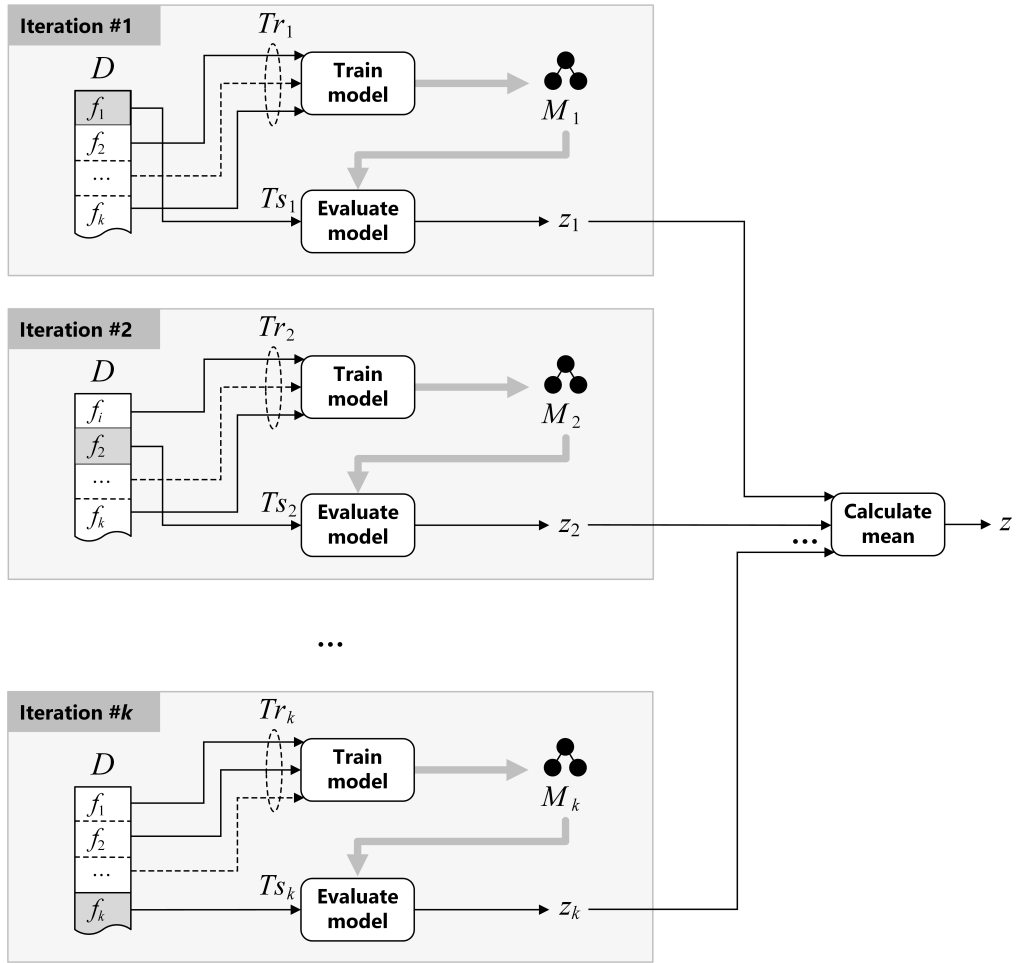
$$Tr_i = D \setminus f_i \quad (2.21) \quad Ts_i = f_i \quad (2.22)$$

Figure 2.3 illustrates the k -fold cross-validation method. In each iteration i , fold i is selected as the test set, and the training set is formed from the examples in the remaining folds. Model M_i is then trained. Similar to the holdout method, any metric can be used for evaluation, resulting in performance z_i . To obtain the final performance estimate, the average of all estimates from each iteration is calculated, resulting in z . In practice, this procedure results in k versions of model M (one for each iteration). The model used for prediction would be a model trained using the entire dataset.

The k -fold cross-validation method has a number of variants, each addressing specific scenarios. *Leave-One-Out Cross-Validation* (LOOCV) (Molinaro et al., 2005), in which k is set to the number of examples in the dataset. In each iteration, only one example is excluded for testing (the test set), while the remaining examples form the training set. *Stratified Cross-Validation* (Kohavi, 1995) ensures balanced class distribution within each fold by creating stratified folds that mirror the class distribution of the entire dataset. *Repeated Cross-Validation* (Zhong et al., 2020) involves replicating the k -fold process multiple times, enhancing the reliability of the estimated generalization error. *Bootstrapping* (Kohavi, 1995) uses random sampling with replacement to generate multiple subsets, offering a robust estimation of generalization error.

2.2 Rule-based models and related formats

Rule-based systems have been a focus of machine learning research since the early 1960s. Their arose from constructing knowledge-based systems capable of automated operation (Fürnkranz et al., 2012). The core objective of such systems is to infer new facts based on the given facts and predefined rules that describe the problem domain. A knowledge base is a formal model of a specific domain, consisting primarily of facts and rules (Sommer, 1996). By the mid-1980s, rule-based systems had already established themselves as a practical tool for automating problem-solving expertise, capturing and refining human knowledge, and demonstrating commercial viability (Hayes-Roth, 1985). Despite being considered the simplest form of artificial intelligence (Grosan and Abraham, 2011), their relevance has persisted into modern times as a robust strategy for knowledge representation and developing components for decision-making support (Ligęza, 2006).

Figure 2.3: Estimation of performance with the k -fold cross-validation method

Although frequently used interchangeably, there is a clear distinction between rule-based systems and rule-based models. *Rule-Based Systems* rely on a predefined set of human-coded instructions or rules that dictate their behavior and lead to predetermined outcomes (Grosan and Abraham, 2011). Essentially, they operate based on the knowledge explicitly encoded by human experts. In contrast, *Rule-based models* use machine learning algorithms to automatically learn rules from a dataset without requiring explicit human intervention. Rule-based models have been extensively incorporated as knowledge representation into expert systems (Langley and Simon, 1995; Rowley, 2007), establishing a natural connection to human reasoning and improving understanding and explanation of other, less readily interpretable models (Barredo Arrieta et al., 2020). The set of rules showcases the relationships between the values of independent variables to predict the value of a target variable.

A *rule* comprises two components: the *preconditions*, also often referred to by multiple terms as the *antecedent*, *premise*, *body*, or *left hand side*, and the *conclusion*, also known as *consequent*, *action*, *head*, or *right hand side* (Ligeza, 2006; Alonso Moral, 2007; Grosan and Abraham, 2011). In propositional logic, a rule is represented as a function between the preconditions and the con-

clusions (Equation 2.23). The preconditions define conditions that must be satisfied for the rule to become applicable or activated, determining the rule's conclusion, which correspond to the effects, decision, or action taken upon activation (Ligeza, 2006). In inductive logic, used by programming languages such as Prolog (Sterling and Shapiro, 1994), a rule is expressed as $Head \leftarrow Body$ or more compactly as $H \leftarrow B$ (Gomes, 2008).

$$r : \langle preconditions \rangle \rightarrow \langle conclusions \rangle \quad (2.23)$$

In the simplest form of rule-based classification models, also known as *first-order learners* (Lavrač et al., 1999), a rule's structure involves a conjunction of propositions as its preconditions, resulting in a class prediction for the target variable $\hat{y}$ as a single conclusion (Equation 2.24). A *proposition*, also known as a *predicate* (Lakkaraju et al., 2016) and denoted as p , is a Boolean statement, which consequently, can only be either true or false (Rosen, 2012).

$$r : p_1 \wedge p_2 \wedge \dots \rightarrow \hat{y} \quad (2.24)$$

Each proposition holds a test or condition applied to the variables' values. These variables typically correspond to the independent variables in the training dataset. The test can involve comparing a variable value to a single value using relational operators like $=$, $<$, $\leq$, $>$, and $\geq$ (Equation 2.25). Alternatively, it can involve checking whether a variable value belongs to a set or interval using the membership operator $\in$ (Equations 2.26 and 2.27).

$$p = x \langle relational\ operator \rangle v \quad (2.25)$$

$$p = x \in \{c_1, c_2, \dots, c_j\} \quad (2.26)$$

$$p = x \in [a; b] \quad (2.27)$$

According to the concept of *internal disjunction* (Ligeza, 2006), any disjunction of equalities can be transformed into membership (Equation 2.28) in a subset t (Equation 2.29). A *subset*, by definition, is always a subset of the domain of the tested variable ($t \subseteq \text{dom } x$). A similar extension applies to interval representation. For instance, $x \in [v_1; v_2]$ indicates that all values within the interval $[v_1; v_2]$ are potential values for x (applicable to both discrete and continuous domains of the variable). Therefore, in the case of equality and membership, the concept can be generalized, and the notation abstracted to express a proposition as a variable testing membership in a subset (Equation 2.30).

$$[(x = v_1) \vee (x = v_2) \vee \dots \vee (x = v_j)] \equiv [x \in t] \quad (2.28)$$

$$t = \{v_1, v_2, \dots, v_j\} \quad (2.29)$$

$$p = x \in t \quad (2.30)$$

In the case of infinite, countable, and continuous domains, the above transformation no longer applies (Ligeza, 2006). This indicates that inequalities involving numerical variables, such as the proposition $p = x \leq v_1$, would correspond to the subset notation $p = x \in (-\infty, v_1]$. Since infinite

If the value of x_1 is at least a_1 but less than a_2 and the value of x_1 is less than b_1 , the decision is c_2 . Additionally, if the value of x_1 is still at least a_1 though less than a_3 and the value of x_2 is at least b_1 but no greater than b_2 , the decision is also c_2 . In any other circumstance, the decision is c_1 .

Figure 2.4: A colloquial description of a rule-based model

bounds are not admissible because they represent an infinite internal disjunction, the transformation cannot be used directly. However, it can be adapted by selecting a minimum bound l_x from the domain of x , specifically $p = x \in (l_x, v_1]$, with $l_x \in \text{dom } x$. It is important to note that the lower bound becomes open. Similarly, if $p = x < v_1$, the upper bound is also open, with $p = x \in (l_x, v_1)$. The same principle applies to positive upper bounds.

Within a rule, propositions are the elements of the rule's *itemset*, denoted by s (Equation 2.31). Despite other possible representations (propositional or inductive logic), a rule represented as r , can be formally condensed into a tuple (Equation 2.32), encompassing both the itemset and the predicted class label (Lakkaraju et al., 2016). Formally, a rule-based model is a *rule-set*, denoted by $\mathcal{R}$, encompassing R distinct rules (Equation 2.33).

$$s = \{p_1, p_2, \dots\} \quad (2.31) \quad r = (s, \hat{y}) \quad (2.32)$$

$$\mathcal{R} = \{r_1, r_2, \dots, r_R\} \quad (2.33)$$

Rule-based models can be presented in various forms (Ligeza, 2006). To provide a clear illustration, Figure 2.4 presents a description of a simple rule-based model using everyday language, with only two numerical variables (x_1 and x_2) and corresponding decisions (classes c_1 and c_2). The following sections introduce and describe alternative formats for presenting the same model.

2.2.1 Decision trees

Decision trees are predictive models resembling flowcharts where nodes evaluate conditions on independent variables to make decisions (Han et al., 2012). Structurally, a decision tree encompasses a *root node*, which is the starting point of the tree, from which branches diverge. The *leaf nodes* are terminal nodes representing the final decisions or predictions, and the *intermediate nodes* are non-terminal nodes with conditions that lead to further branching and sub-trees.

Each non-leaf node within the tree holds a condition that involves a *splitting variable*, which is the independent variable being evaluated at the node, a *relational operator* used to compare the variable's value, and a *split point*, a specific value or set against which the variable is compared. Depending on whether the condition is evaluated as true or false, the flow of the decision tree follows the corresponding branch, eventually reaching a leaf node and its associated prediction.

Decision tree construction involves two characteristics of node splitting: the number of variables tested and the number of branches per node. *Univariate splits* test only one variable at each

node, offering simplicity but potentially missing complex interactions. *Multivariate splits* consider multiple variables simultaneously, capturing combined relationships but increasing model complexity (Han et al., 2012). Regarding the number of branches per node, *binary splits* generate two child nodes per node based on a condition, resulting in simpler binary decision trees. *Multiway splits* allow for more than two branches, offering greater flexibility but potentially increasing computational cost (Kim and Loh, 2001). Choosing the appropriate alternatives depends on dataset characteristics, desired model complexity, and available resources.

Figure 2.5 depicts a binary decision tree model corresponding to the rule-based model presented in Figure 2.4. Each incoming example, characterized by values for independent variables (x_1 and x_2), traverses the tree based on the outcome of condition evaluations at each node. The initial evaluation occurs at the root node. If the condition involving x_1 is satisfied, the flow follows the *yes* branch and predicts c_1 for the target variable. Conversely, if the condition is not met, the flow proceeds to the *no* branch towards the next node. This process of evaluating conditions and following corresponding branches continues until a leaf node is reached, with each unique path from the root to a leaf representing a rule within the decision tree (Guidotti et al., 2018).

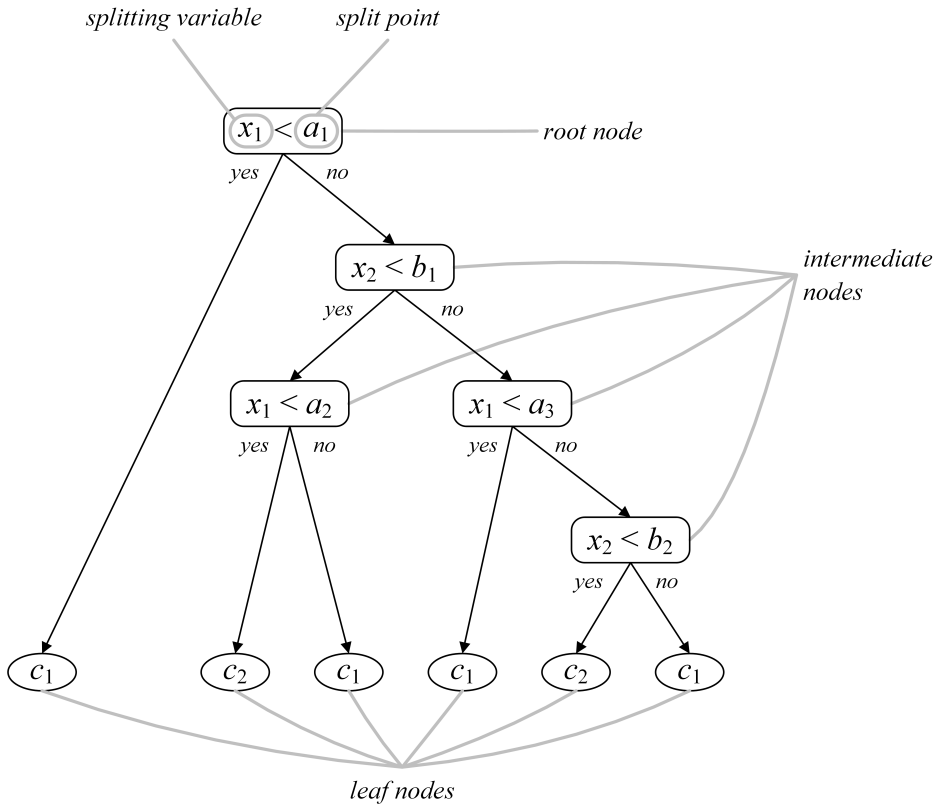


Figure 2.5: A rule-based model represented as a binary decision tree

There are multiple algorithms for inducing decision trees, the most pivotal being *Classification And Regression Trees* (CART) (Breiman et al., 1984) and *C5.0* (Kuhn and Quinlan, 2023), an evolution of *Iterative Dichotomiser 3* (ID3) (Quinlan, 1986). Subsequently other algorithm were developed such as *M5'* (Witten, 1997) and *Logistic Model Trees* (LMT) (Landwehr, 2005). These

Listing 2.1: A binary decision tree generated by the C5.0 algorithm

```
x1 < a1: c1
x1 >= a1:
: ... x2 < b1:
:   : ... x1 < a2: c2
:   :   x1 >= a2: c1
:   x2 >= b1:
:   : ... x1 < a3: c1
:   :   x1 >= a3:
:   :   : ... x2 < b2: c2
:   :   :   x2 >= b2: c1
```

algorithms typically follow a greedy, top-down, recursive divide-and-conquer approach. Beginning with the whole training set of examples and their corresponding class labels, these algorithms iteratively partition the data into smaller subsets as the tree grows (Han et al., 2012). This process continues until a stopping criterion is met, resulting in a tree that captures the relationships between variables and the target variable.

Regardless of the chosen algorithm, decision trees benefit from two properties: mutual exclusivity and exhaustiveness (Han et al., 2012; Guidotti et al., 2018). Mutual exclusivity ensures that each example falls under a unique rule, preventing redundancy and ambiguity. This is achieved by assigning each leaf node to a distinct rule. Exhaustiveness, conversely, guarantees the tree covers the entire decision space. A corresponding rule always applies to every possible combination of variable values, satisfying the completeness property.

Beyond graphical representations, decision trees can also be expressed in a textual format. For instance, the *R statistical computing environment* (R Core Team, 2023) offers packages specifically designed for decision tree algorithms, which often generate textual representations of the trees (Kuhn and Quinlan, 2023; Therneau et al., 2023). These textual representations typically span multiple lines, with each line corresponding to a node in the tree. Indentation is used to depict the hierarchical structure, and various symbols differentiate between different node types. Listing 2.1 exemplifies the decision tree of Figure 2.5 generated by the C5.0 algorithm using the R language.

2.2.2 Binary decision diagrams

Binary decision diagrams (Akers, 1978; Bryant, 2018) are data structures that represent Boolean functions as directed acyclic graphs (Thulasiraman and Swamy, 1992). Similar to a decision tree, a binary decision diagram consists of three types of nodes: a root node where evaluation begins, decision nodes with two outgoing edges based on the condition being true or false (represented by solid and dashed lines, respectively) for subsequent conditions, and terminal nodes holding the final decisions (Ligeza, 2006).

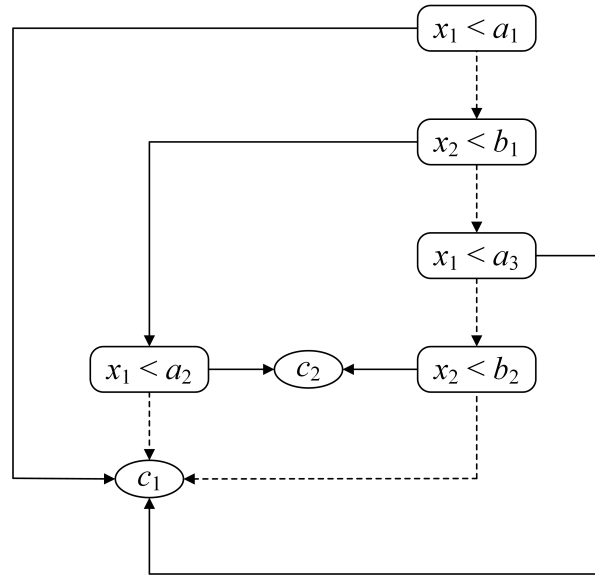


Figure 2.6: A rule-based model represented as a binary decision diagram

In addition to their distinct notation, binary decision diagrams offer an advantage over decision trees: their reduced structure (Somenzi, 1999). Reduction is achieved by eliminating redundant subtrees and ensuring each leaf node appears only once. The process involves iteratively applying two strategies. The first is to merge identical nodes, i.e., nodes with the same class (terminal nodes) or identical children (internal nodes) are merged. The second strategy involves removing redundant internal nodes. If an internal node has identical children, it is removed, and its incoming edges are redirected to the shared child. This process continues until no further reductions are possible.

Figure 2.6 depicts a binary decision diagram corresponding to the rule-based model presented in Figure 2.4. In this case, there are no repeated subtrees, so the differences compared to the decision tree in Figure 2.5 are exclusively in the notation of branches and the leaves, where each class is represented only once.

2.2.3 Classification rules

Classification rules form the cornerstone of rule-based models, acting as a set of logical statements or conditions that dictate how data should be categorized into distinct classes. These rules represent a core set, often derived from other formats through conversion processes. A decision tree can be directly transformed into classification rules (Quinlan, 1987a,b). Considering the properties of exclusivity and exhaustiveness, rules directly extracted from a decision tree guarantee that each example is assigned to a single, specific leaf node (Han et al., 2012). This stems from the fact that each path from the root to a leaf in the tree effectively defines a rule. Consequently, the order in which these rules are presented becomes irrelevant. Such rules derived directly from a decision tree are a *decision set* (Lakkaraju et al., 2016).

$$\begin{aligned}
r_1 : x_1 < a_1 &\longrightarrow c_1 \\
r_2 : x_1 \geq a_1 \wedge x_1 < a_2 \wedge x_2 < b_1 &\longrightarrow c_2 \\
r_3 : x_1 \geq a_1 \wedge x_1 \geq a_2 \wedge x_2 < b_1 &\longrightarrow c_1 \\
r_4 : x_1 \geq a_1 \wedge x_1 < a_3 \wedge x_2 < b_1 &\longrightarrow c_1 \\
r_5 : x_1 \geq a_1 \wedge x_1 \geq a_3 \wedge x_2 < b_1 \wedge x_2 < b_2 &\longrightarrow c_2 \\
r_6 : x_1 \geq a_1 \wedge x_1 \geq a_3 \wedge x_2 < b_1 \wedge x_2 \geq b_2 &\longrightarrow c_1
\end{aligned}$$

Figure 2.7: A rule-based model represented as a decision set

While any decision tree can be converted into a set of rules, the reverse is not always true (Margot and Luta, 2021). This underscores the greater expressive power of classification rules. In contrast to the hierarchical structure of decision trees, rule sets can accommodate situations where an example fulfills multiple rules. However, this also creates the possibility of conflicting class assignments (Bramer, 2007; Clark and Boswell, 1991; Otero and Freitas, 2013). To address this problem, many rule induction systems have the rules laid out in a *decision list* (Rivest, 1987), enforcing an order by which the rules are tested. In the decision-making process, an example is presented to each rule sequentially. If the preconditions of a rule are met, the corresponding class prediction is assigned, and the remaining rules are not tested.

Figure 2.7 illustrates the formal rules derived from the natural language description in Figure 2.4. Each rule corresponds to a unique path through the decision tree depicted in Figure 2.5. As an example travels down the tree from the root to a leaf node, the rule's preconditions are formed by the conjunction of propositions encountered at each traversed node (Lior and Maimon, 2008). The conclusion of the rule, then, is the class associated with the corresponding leaf node.

Analyzing the decision tree reveals dependencies between rules, enabling model simplification. Rules r_2 and r_3 activate only if $a_2 \geq a_1$, while r_4 , r_5 , and r_6 require $a_3 \geq a_1$. Additionally, activation of r_5 and r_6 implies that $b_2 \geq b_1$. The decision tree, however, offers no insight into the relationship between a_2 and a_3 . These dependencies and the transitivity principle allow for establishing equivalences that simplify the model (Equation 2.34). This simplification leads to the revised decision set presented in Figure 2.8.

$$\begin{aligned}
x_1 \geq a_1 \wedge x_1 \geq a_2 \wedge a_2 \geq a_1 &\Leftrightarrow x_1 \geq a_2 \\
x_1 \geq a_1 \wedge x_1 \geq a_3 \wedge a_3 \geq a_1 &\Leftrightarrow x_1 \geq a_3 \\
x_2 \geq b_1 \wedge x_2 \geq b_2 \wedge b_2 \geq b_1 &\Leftrightarrow x_2 \geq b_2
\end{aligned} \tag{2.34}$$

Classification rules often use a canonical form to express decision logic and trigger corresponding actions based on the outcome. This notation aligns with conditional structures prevalent in programming languages, making the model essentially a sequence of IF-THEN statements that provide a structured representation of the decision-making process (Grosan and Abraham,

$$\begin{aligned}
r_1 : x_1 < a_1 &\longrightarrow c_1 \\
r_2 : x_1 \geq a_1 \wedge x_1 < a_2 \wedge x_2 < b_1 &\longrightarrow c_2 \\
r_3 : x_1 \geq a_2 \wedge x_2 < b_1 &\longrightarrow c_1 \\
r_4 : x_1 \geq a_1 \wedge x_1 < a_3 \wedge x_2 < b_1 &\longrightarrow c_1 \\
r_5 : x_1 \geq a_3 \wedge x_2 \geq b_1 \wedge x_2 < b_2 &\longrightarrow c_2 \\
r_6 : x_1 \geq a_3 \wedge x_2 \geq b_2 &\longrightarrow c_1
\end{aligned}$$

Figure 2.8: A rule-based model represented as a simplified decision set

2011; Hayes-Roth, 1985). Figure 2.9 presents the rules from Figure 2.8 in a canonical form, often expressed as IF-THEN statements. It can be further simplified by including an ELSE clause indicating the action when the condition evaluates to false.

There are multiple algorithms for classification rule learning, each with its own approach to rule generation. Among these algorithms are PRISM (Cendrowska, 1987), *OneR* (Holte, 1993), *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) (Cohen, 1995), *First Order Regression* (FORS) (Karalič and Bratko, 1997), *RuleFit* (Friedman and Popescu, 2008), *Ensemble of Decision Rules* (EnDer) (Dembczyński et al., 2010), *Node Harvest* (Meinshausen, 2010), *M5 Rules* (Duggal and Singh, 2012), *Stable and Interpretable RULe Set* (SIRUS) (Bénard et al., 2021), and the *Coverage Algorithm* (Margot and Luta, 2021).

2.2.4 Decision tables

Decision tables provide a structured representation of exhaustive and mutually exclusive conditional expressions within specific problem domains (Vanthienen and Wets, 1994). One approach is the *Single-hit decision table* (Huysmans et al., 2011; Lima et al., 2009). This format divides the table into two sections: the condition section listing variables and their values and the decision section presenting the mutually exclusive decisions for each rule. Each column, excluding the

$$\begin{aligned}
&\text{IF } x_1 < a_1 \text{ THEN } \hat{y} = c_1 \\
&\text{IF } x_1 \geq a_1 \text{ AND } x_1 < a_2 \text{ AND } x_2 < b_1 \text{ THEN } \hat{y} = c_2 \\
&\text{IF } x_1 \geq a_2 \text{ AND } x_2 < b_1 \text{ THEN } \hat{y} = c_1 \\
&\text{IF } x_1 \geq a_1 \text{ AND } x_1 < a_3 \text{ AND } x_2 < b_1 \text{ THEN } \hat{y} = c_1 \\
&\text{IF } x_1 \geq a_3 \text{ AND } x_2 \geq b_1 \text{ AND } x_2 < b_2 \text{ THEN } \hat{y} = c_2 \\
&\text{IF } x_1 \geq a_3 \text{ AND } x_2 \geq b_2 \text{ THEN } \hat{y} = c_1
\end{aligned}$$

Figure 2.9: A rule-based model represented as a decision set in the canonical form

first (variable names), represents a single rule, and a dashed symbol (-) in the condition section indicates the corresponding value is irrelevant for that specific rule. The predicted class for each rule is marked with a $\times$ symbol.

Table 2.4 exemplifies this approach by representing the model from Figure 2.4 as a single-hit decision table. Single-hit decision tables offer a structured way to define rule sets, but creating them directly from the rules can lead to large tables with many columns. Table contraction methods can be employed to address this issue (Freitas, 2014). These methods aim to generate a compact decision table with an optimal condition order and minimal size while ensuring it remains logically equivalent to the original rule set.

Table 2.4: A rule-based model represented as a single-hit decision table

x_1	$< a_1$	$< a_2$	$\geq a_2$	$< a_3$	$\geq a_3$	
x_2	-	$< b_1$	$< b_1$	$\geq b_1$	$< b_2$	$\geq b_2$
c_1	$\times$		$\times$	$\times$		$\times$
c_2		$\times$			$\times$	

The *Extended attributive decision table* offers an alternative format to representing rule-based models in a tabular format (Ligeza, 2006), unlike the single-hit format, where each column represents a rule, extended attributive tables present variables in columns and rules in rows. However, they differ in expressing propositions within rules, using set membership notation ($p : x \in t$). Sets, defined as intervals for numerical variables and enumerated sets for categorical variables, form the foundation for evaluating variable values. This notation signifies that each variable's value is assessed against a specific subset of its domain, denoted as $t_{i,j}$ for variable j in rule i . Consequently, each rule is represented as a conjunction of set memberships (Equation 2.35). This format offers a more concise and flexible way to define rules than the single-hit format, mainly when dealing with complex relationships between variables and their values. Additionally, it allows for direct translation between decision trees, decision sets, and decision tables, improving versatility in rule-based modeling.

$$r_i : (x_1 \in t_{i,1}) \wedge (x_2 \in t_{i,2}) \wedge \cdots \wedge (x_d \in t_{i,d}) \longrightarrow \hat{y}_i \quad (2.35)$$

The extended attributive format applied to the example model in Figure 2.4 is presented in Table 2.6. Since the variable domains are unspecified in Figure 2.4, negative and positive infinity are assumed as lower and upper bounds for x_1 and x_2 , respectively. However, the internal disjunction (Equation 2.28) is incompatible with infinite bounds. Therefore, these bounds are replaced with specific values denoted as l_x and u_x , likely determined by domain expertise. Empty cells in the table, like $t_{1,2}$, indicate that the corresponding variable is not tested in that rule, making the rule applicable for any variable value. This is formally equivalent to setting the domain of the variable, $t_{1,2} = \text{dom}.x_2$. As infinite limits are not allowed, the cell could be filled as $t_{1,2} = (l_{x_2}; u_{x_2})$.

Table 2.5: Structure of an extended attributive decision table

Rule	x_1	x_2	$\dots$	x_d	$\hat{y}$
r_1	$t_{1,1}$	$t_{1,2}$	$\dots$	$t_{1,d}$	$\hat{y}_1$
r_2	$t_{2,1}$	$t_{2,2}$	$\dots$	$t_{2,d}$	$\hat{y}_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_R	$t_{R,1}$	$t_{R,2}$	$\dots$	$t_{R,d}$	$\hat{y}_R$

Table 2.6: A rule-based model represented as an extended attributive decision table

Rule	x_1	x_2	$\hat{y}$
r_1	$(l_{x_1}; a_1)$		c_1
r_2	$[a_1; a_2)$	$(l_{x_2}; b_1)$	c_2
r_3	$[a_2; u_{x_1})$	$(l_{x_2}; b_1)$	c_1
r_4	$[a_1; a_3)$	$(l_{x_2}; b_1)$	c_1
r_5	$[a_3; u_{x_1})$	$[b_1; b_2)$	c_2
r_6	$[a_3; u_{x_1})$	$[b_2; u_{x_2})$	c_1

2.3 Summary

This chapter is the groundwork for the research presented in this thesis. It introduces the fundamental concepts of knowledge discovery, data mining, and machine learning, establishing the theoretical context for this research. Subsequently, the chapter presents consistent terminology and notation about data representation and related concepts. This includes a discussion of examples, variables, variable domains, and the representation of datasets. Additionally, the chapter explores training predictive models, a critical step in extracting knowledge from data, and methods to estimate their predictive performance.

The second part of the chapter focuses on rule-based models, a specific class of models central to this investigation. It presents the most prevalent notation in this field and provides a comprehensive overview of various rule-based model formats. These formats are clarified through both descriptive explanations and illustrative examples. Particular emphasis is placed on decision trees, a popular rule-based model format known for its high interpretability. Furthermore, it discusses classification rules, another standard format for representing rule-based models. It also explores decision tables, particularly the extended attributive decision table variant, the format used in this research.

Chapter 3

Related Work

To contextualize the contributions of this thesis, this chapter reviews the related literature. It is structured in two sections. The first section provides an overview of model interpretability, including a taxonomy of techniques, an analysis of evaluation methods, and a discussion of current challenges. The second section examines approaches for merging prediction models, distinguishing them from ensemble learning and further exploring underlying analytical and mathematical approaches.

3.1 Model interpretability

Black-box models, also referred to as *opaque models* (Johansson et al., 2006) or *complex models* (Bastani et al., 2017), are characterized by the inherent difficulty or even impossibility for humans to understand the process by which they generate predictions (Han et al., 2012). This opacity arises from the models' use of highly complex functions or internal logic that remains undisclosed or obscure (Guidotti et al., 2018). Examples of algorithms generating black-box models include *Neural Networks* (NN) (Rumelhart et al., 1986) and *Support Vector Machines* (SVM) (Steinwart and Christmann, 2008). This lack of transparency restrains users, both experts and non-experts, from verifying and comprehending the reasoning and decision-making processes underlying the system's predictions. As depicted in Figure 3.1, the black-box metaphor signifies machine learning models that receive input, a new example, and produce an output, i.e., a prediction. Still, the internal process leading to the prediction remains inaccessible (Carvalho et al., 2019).

White-box models, also referred to as *transparent models* (Barredo Arrieta et al., 2020) or *interpretable models* (Lisboa et al., 2023), are characterized by their readily comprehensible prediction generation process for humans (Linardatos et al., 2021). This transparency arises from the models' reliance on simple functions or inspectable internal logic (Guidotti et al., 2018). Linear and rule-based models (decision trees and classification rules) are commonly recognized as white-box models. The white-box metaphor, therefore, represents models with transparent inner

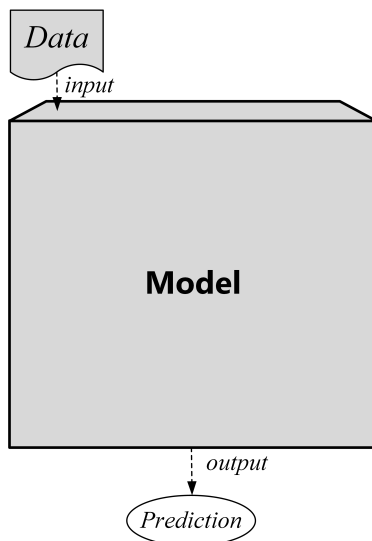


Figure 3.1: Black-box model

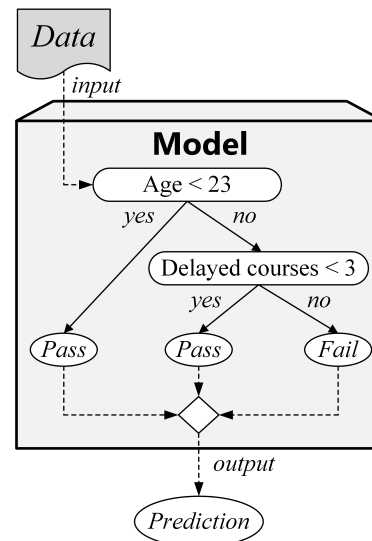


Figure 3.2: White-box model

workings, enabling users to trace the logic from input to output. As illustrated in Figure 3.2, an input traverses the decision tree’s flow, resulting in a prediction, all within a readily understandable format (Molnar, 2019).

Due to the exponential increase of heterogeneous data collection and massive computational power, machine learning systems have reached greater predictive performance and, for most of them, greater complexity (Carvalho et al., 2019). Deep learning methods exemplify this trend, with architectures like *Deep Residual Networks* (ResNets) (LeCun et al., 2015; He et al., 2016) exceeding 200 layers and have been shown to beat human-level performance on object recognition tasks (Carvalho et al., 2019). Other examples are ensemble methods, such as *Random Forests* (Breiman, 2001), *AdaBoost* (Freund and Schapire, 1997), and *XGBoost* (Chen and Guestrin, 2016), all having demonstrated remarkable predictive performance. A significant body of research is devoted to exploring and addressing these topics.

While ubiquitous, machine learning models often remain opaque to many, especially those lacking specialized data analysis expertise. This presents a notable limitation as the effectiveness of these models relies on their skill to convey knowledge to those outside the field. A vital aspect of this communication is its accessibility to non-experts, ensuring the information is understandable and actionable. Ineffective communication can cause a swift decline in the interest and trust of decision-makers, which can ultimately undermine their faith in predictions and the model’s overall reliability. Trust in individual predictions refers to a user’s confidence in the validity of a specific outcome, whereas trust in the model reflects a user’s belief in its overall reliability and predictive accuracy (Ribeiro et al., 2016).

Model development often faces a trade-off between optimal predictive performance and interpretability (Linaardatos et al., 2021). Therefore, a growing emphasis is placed on creating models that not only accurately predict future events but also reveal the relationships between variables towards the prediction. In other words, a model’s value extends beyond its predictive capabilities. It

should also enable decision-makers to understand the rationale behind its predictions. This understanding fosters user trust, encourages model adoption, and assists developers in comprehending complex issues. This need has driven increasing interest in interpretability from both academia and industry (Du et al., 2020).

3.1.1 Interpretability and explainability

Interpretability lacks a universally accepted definition (Guidotti et al., 2018). Common formulations emphasize a human’s ability to predict model outcomes, focusing on understanding how a model arrives at its results (Kim et al., 2016); to explain model behavior by translating its internal workings into human-understandable terms (Doshi-Velez and Kim, 2017); and to comprehend the decision rationale, understanding the factors influencing the model’s decisions (Miller, 2017). The inherent subjectivity of interpretability, encompassing various aspects rather than a single concept, makes it hard to come up with a universal formalization (Rüping, 2006) possibly due to the lack of a mathematical definition (Lipton, 2018; Molnar, 2019; Carvalho et al., 2019). Additionally, some authors argue for a domain-specific understanding of interpretability, suggesting a unique definition might not be feasible (Freitas, 2014; Rudin, 2019). Therefore, interpretability, in its standalone form, is a broad and often loosely defined concept. In its most general sense, it encompasses extracting information from data in various forms to foster the comprehension of a model’s behavior (Murdoch et al., 2019; Gilpin et al., 2018).

Explainability is sometimes used as a synonym for interpretability, reflecting their close relationship (Linardatos et al., 2021; Silva et al., 2018; Du et al., 2020; Miller, 2017). However, some distinctions have been proposed. One perspective views interpretability as enabling human comprehension of the relationship between a model’s inputs and outputs, while explainability delves deeper into the internal workings and decision-making processes (Linardatos et al., 2021). Another suggests that interpretability is the initial step towards understanding, with explainability providing a more comprehensive breakdown of the model’s inner mechanisms, fostering trust and ensuring sound decision-making, particularly for black-box models (Gilpin et al., 2018). Nevertheless, some argue for their interchangeability, suggesting that an interpretable model inherently obviates the need for further explanation, and vice versa (Leblanc and Germain, 2023).

While seemingly overlapping, interpretability and explainability have fostered distinct research areas. *Interpretable Machine Learning* (IML) emerges as a multidisciplinary field requiring collaboration between computer science, human-computer interaction, and social science. Its central aim lies in developing user-centric and human-friendly explanations for machine learning models (Du et al., 2020; Molnar, 2019). In contrast, *eXplainable Artificial Intelligence* (XAI) addresses the broader need for accountable, fair, and high-performing models in real-world applications (Ali et al., 2023). It emphasizes understanding AI systems’ behavior and decision-making processes, shifting the focus from only optimizing for predictive accuracy (Linardatos et al., 2021). However, some authors argue that XAI lacks a universally accepted definition (Adadi and Berrada, 2018). Instead, it can be viewed as a movement, initiative, and ongoing effort to address concerns

about transparency and trust in artificial intelligence rather than a strictly technical concept (Gunning et al., 2021).

The term XAI gained significant momentum with the introduction of the European Union’s *General Data Protection Regulation* (GDPR), which emphasizes the need for AI systems to be explainable and support the decision-making processes (European Parliament, 2016; The Royal Society, 2019). This surge in interest aligns with growing concerns surrounding the lack of an explicit *Right to Explanation* (Goodman and Flaxman, 2016; Wachter et al., 2016), ensuring diversity and inclusion in AI, mitigating the potential for bias and inequality in automated systems, and fostering the development of secure AI for safety-critical applications.

Both interpretability and explainability have a wide range of applications, with the literature highlighting three areas: model validation, model debugging, and knowledge discovery. Model validation involves assessing whether models rely on sound evidence or biases present in training data. For example, explanations can uncover gender and ethnic biases, ensuring ethical compliance. Model debugging entails analyzing misbehavior and erroneous predictions, often stemming from adversarial learning. Explanations facilitate identifying model flaws and exploring potential solutions. Knowledge discovery enables humans to understand model decision-making, leading to domain expert feedback from data (Du et al., 2020).

While interpretability and explainability are highly sought-after qualities in machine learning, it is essential to acknowledge that not all systems require them. Detailed explanations become less necessary in low-stakes scenarios where errors have minimal impact or in well-studied and validated problems where there is confidence in the model’s decisions despite their opaqueness. However, when the problem is incompletely understood or formalized, hindering optimization and evaluation, interpretability and explainability become paramount (Doshi-Velez and Kim, 2017).

3.1.2 Desiderata of interpretable models

When evaluating interpretable models, a set of desiderata, or desired characteristics, are considered (Guidotti et al., 2018; Molnar, 2019). Although the focus is on interpretability, predictive accuracy remains an important desideratum. Accuracy, in this context, refers to the model’s ability to correctly predict outcomes for unseen examples (Guidotti et al., 2018; Lavrač et al., 1999). While the standard accuracy metric is often used, a wider range of metrics may be necessary to fully evaluate interpretable models. Therefore, the goal is to develop models that are both interpretable and maintain competitive levels of predictive accuracy.

The dilemma between accuracy and interpretability is a well-known issue in machine learning (Lundberg and Lee, 2017), often framed as a trade-off between these two qualities (Rüping, 2006; Flora et al., 2022a). This manifests as a dichotomy: complex models often achieve high performance but are difficult to interpret, while simpler models prioritize interpretability at the expense of state-of-the-art accuracy (Linardatos et al., 2021). This situation originates from the imperative to balance *predictive accuracy* with *descriptive accuracy*, the latter being the *fidelity* with which an explanation method captures the relationships learned by the underlying model (Murdoch et al., 2019). Achieving high fidelity in explanations is critical, as it ensures their accurate

representation of the model’s internal workings and decision-making processes, thereby enabling users the ability to assess the validity and reliability of the explanations. Consequently, the objective is to optimize this balance, employing appropriate metrics to evaluate both predictive and descriptive performance (Su et al., 2016).

In addition to predictive accuracy, interpretable models also encompass other desiderata. Ethical considerations are paramount; interpretability promotes fairness by enabling the detection and mitigation of biases that might otherwise remain hidden in opaque models (Hardt et al., 2016). This transparency also safeguards against discrimination by allowing scrutiny of the model’s decision-making process to ensure unbiased outcomes (Romei and Ruggieri, 2013). Furthermore, interpretability plays a vital role in protecting privacy, particularly given the potential for complex models to inadvertently learn and reveal sensitive patterns within data (Aldeen et al., 2015). Adhering to privacy standards, exhibiting non-discriminatory behavior, and complying with user-defined constraints like monotonicity are essential for building trust in interpretable models. These factors align with users’ expectations of causal relationships between inputs and outputs, fostering confidence in the model’s reliability by allowing users to understand its behavior under controlled scenarios (Martens et al., 2011; Verbeke et al., 2011; Guidotti et al., 2018).

3.1.3 Interpretability taxonomy overview

The literature recognizes three key criteria for interpretability: scale, method, and specificity. These criteria are used within a taxonomy for classifying and understanding various interpretability techniques (Molnar, 2019). This taxonomy is represented in Figure 3.3, with an overview of the categories within each criterion, along with the most prevalent approaches and illustrative methods or techniques.

Scale, in the context of interpretability, refers to the extent of the prediction process that interpretability methods aim to explain (Carvalho et al., 2019). *Global interpretability* is the ability to comprehend the entire logic of a model by understanding the complete reasoning process that leads to all potential outcomes (Guidotti et al., 2018). *Local interpretability*, in contrast, involves understanding the reasons for a specific decision (Doshi-Velez and Kim, 2017). These two perspectives are not mutually exclusive; each offers distinct advantages. Global interpretability promotes transparency and enhances user understanding. Local interpretability uncovers causal relationships between specific examples and corresponding model predictions, clarifying the model’s decision-making process. Thus, global and local interpretability complement each other, fostering trust in the model’s overall performance and individual predictions, respectively (Du et al., 2020). Global interpretability may be necessary when seeking scientific understanding or bias detection, while local interpretability is often essential for supporting a specific decision (Doshi-Velez and Kim, 2017).

The *method* of interpretability refers to when interpretability is obtained, categorized as in-model, post-model, or pre-model (Molnar, 2019). *In-model interpretability*, also referred to as *ante-hoc*, refers to inherently interpretable machine learning models (Burkart and Huber, 2021; Lisboa et al., 2023), often achieved through their intrinsic design (i.e., white-box models) or

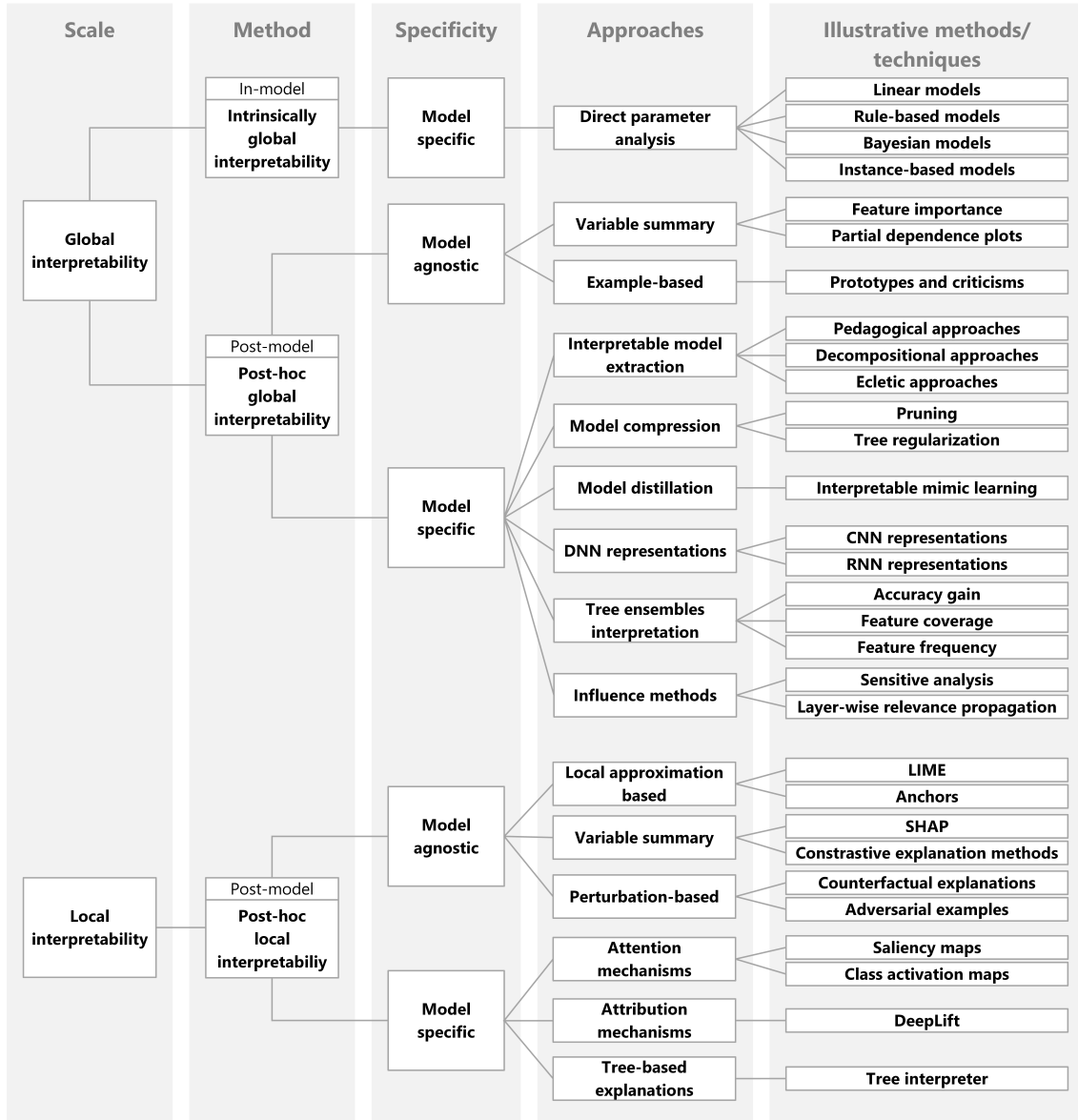


Figure 3.3: Interpretability taxonomy overview

by incorporating specific constraints. *Post-model interpretability*, in contrast, typically involves augmenting black-box models with an interpretability layer after training, often employing a secondary model (the *post-hoc* model) to explain or interpret the existing one (Carvalho et al., 2019; Molnar, 2019). The primary distinction between these two methods lies in the trade-off between model accuracy and explanation fidelity. While in-model approaches may offer precise and clear explanations, they can compromise predictive performance; conversely, post-hoc methods preserve the underlying model’s accuracy but are limited by their approximative nature (Du et al., 2020). *Pre-model interpretability*, a third method recognized by some authors, focuses on understanding the data itself, independent of any specific model and prior to model selection, thus closely relating to data interpretability (Carvalho et al., 2019).

Specificity, in the context of interpretability, refers to the relationship between the interpretability method and the machine learning model it can be applied to (Molnar, 2019), an important factor when selecting an appropriate technique. Methods can be categorized as model-specific or model-agnostic. *Model-specific methods* are tailored to particular model types, such as decision trees, capitalizing on their internal mechanisms to generate explanations; for instance, decision tree techniques use the tree structure to elucidate the impact of specific variables on predictions. While effective for their corresponding models, these methods lack universal applicability. *Model-agnostic methods*, conversely, are designed to be universally applicable, capable of explaining any machine learning model regardless of its underlying algorithm or complexity. However, this adaptability comes at a cost: lacking access to internal model mechanisms, model-agnostic methods often rely on analyzing input-output relationships, potentially leading to less precise explanations compared to model-specific approaches (Carvalho et al., 2019).

Beyond the criteria of scale, method, and specificity, other factors also influence interpretability. *Problem incompleteness*, for instance, requires identifying and understanding gaps in the problem formulation, which can arise from incomplete examples, constraints, or limitations inherent to the domain itself; the severity of this incompleteness dictates the need for explanations, ranging from specific scenario checks to general model property verification (Doshi-Velez and Kim, 2017). *Explanation time sensitivity* is also important, as user time constraints impact the desired level of detail. Time-sensitive situations often necessitate concise explanations, whereas more complex scenarios, such as loan approvals, may allow for more elaborate explanations (Guidotti et al., 2018). Furthermore, *user expertise* influences the evaluation of interpretability; domain experts, for example, might favor larger, more sophisticated models despite potential opacity, due to their ability to navigate them effectively (Grossman et al., 2005).

3.1.4 Evaluation of interpretability

Defining and assessing interpretability in machine learning models is challenging due to the lack of a universally accepted definition. Due to the concept’s inherent flexibility and occasional ambiguity, evaluations have been categorized into three main types: application-grounded, human-grounded, and functionally-grounded (Gilpin et al., 2018). This reflects that interpretability assessments can range from task-specific considerations to broader perspectives.

Application-grounded evaluation, a real-world approach to assess interpretability, involves conducting experiments with humans within a specific task or domain (Doshi-Velez and Kim, 2017). For instance, when developing a model to assist doctors in diagnosing diseases, evaluating its efficacy in a real-world diagnostic setting allows for the most effective demonstration. This aligns with standard human-computer interaction and visualization evaluation methods, emphasizing the importance of successful task fulfillment (Antunes et al., 2012). In this approach, the explanation generated by the model is used in a real-world scenario and evaluated by the end user, who is also a domain expert. A suitable benchmark involves comparing the effectiveness of the explanation to how well humans can explain the same decision (Molnar, 2019).

Human-grounded evaluation offers an alternative when real-world application is challenging (Doshi-Velez and Kim, 2017). This approach involves conducting simplified experiments with human subjects that capture the core aspects of the target application. This method has the advantage of having a broader participant pool. Unlike application-grounded evaluation, it allows participation from non-experts, reducing costs associated with recruiting domain experts (Du et al., 2020). Additionally, it focuses on general aspects as it is suitable for assessing broader explanation quality, such as understandability under time pressure. Researchers can design controlled experiments focusing on specific aspects like explanation format (e.g., text vs. visuals) while controlling external factors such as overall task complexity (Kim et al., 2015; Lakkaraju et al., 2016).

Functionally-grounded evaluation relies on formal definitions of interpretability to indirectly evaluate the quality of explanations (Molnar, 2019). This approach offers significant advantages by eliminating the need for time-consuming and resource-intensive human experimentation. This type of evaluation becomes particularly relevant when a method is still in its early stages or when conducting human-subject experiments would be deemed unethical. In the context of an evaluation using metrics, this approach enables the comparison of interpretability across models. Examples of metrics are explored in the next sections.

3.1.4.1 Functional measures in decision rules

This section presents the key metrics used to evaluate the interpretability of individual rules within rule-based models, focusing on those relevant to the work presented in this thesis. A rule is defined as a tuple $r = (s, \hat{y})$, where s represents the rule’s antecedent, also known as the itemset (Equation 2.31). This itemset is a set of propositions connected by conjunction, forming the conditions that must be met for the rule to apply (Equation 2.31). The term $\hat{y}$ represents the predicted class label assigned by the rule when its conditions are satisfied (Lakkaraju et al., 2016).

Rule’s complexity. *Rule length*, also known as *rule size* or *rule cardinality*, refers to the number of propositions a rule contains (Lakkaraju et al., 2016). These conditions typically involve logical functions, usually considered understandable by humans (Cruz et al., 2015). As the number of propositions in a rule increases, its interpretability tends to decrease. Formally, the rule length for a given rule r is the number of propositions in the rule’s itemset (s) (Equation 3.1).

$$Length(r) = |s| \quad (3.1)$$

Rule sparsity involves the number of variables used in a rule. This concept is analogous to *sparse data*, where datasets contain a high percentage of missing values (Petrovski and Bizer, 2016). In this analogy, a sparser rule uses fewer variables, effectively missing more from the complete set of available variables. Formally, the sparsity of a rule r is defined as the complement of the ratio between the number of variables in the rule ($|V_r|$) and the total number of independent variables in the dataset ($|X|$) (Equation 3.2), from which the model was trained. However, the

definition of sparsity in machine learning is not universally agreed upon, leading to different interpretations (Hurley and Rickard, 2009). There are studies where sparsity is a synonym with rule length (Feldman, 2000; Su et al., 2016), while in others, sparsity refers to the number of unused variables by a model (Rüping, 2006).

$$Sparsity(r) = 1 - \frac{|V_r|}{|X|} \quad (3.2)$$

Rule length and rule sparsity offer complementary perspectives on a rule’s complexity. Rule length indicates the total number of conditions, reflecting the rule’s overall intricacy. Rule sparsity, on the other hand, emphasizes variable usage; fewer variables generally suggest higher interpretability. These two metrics are not directly related. A rule can be sparse, using a limited subset of available variables, even with a large number of conditions. Thus, a rule can have many conditions yet still be considered sparse due to its limited variable usage.

Rule’s coverage. *Coverage*, in the context of rule-based models, quantifies the relationship between a rule and the data it applies to. The term is used both to describe a set of examples and as a metric. The *rule coverage set* for a given rule, r , is the set of examples within a dataset that satisfy all the rule’s conditions (propositions) (Lakkaraju et al., 2016). Formally, this set comprises a subset of examples from the dataset, where each example is a tuple of values, $e_i = (v_{i,1}, v_{i,2}, \dots, v_{i,d})$. An example is included in the rule coverage set only if all its independent variable values ($v_{i,j}$) simultaneously satisfy the constraints specified by the rule’s propositions, implying membership in the corresponding subsets (t_j) (Equation 3.3). Variables without associated propositions are disregarded. Building upon this concept, *rule coverage*, as a metric, measures the proportion of examples in the entire dataset (of size n) that fulfill the rule’s conditions (Equation 3.4). In other words, an example is considered covered by a rule when its independent variable values satisfy all the rule’s propositions (Lavrač et al., 1999). Higher coverage generally indicates a more general rule. It is important to note that when determining coverage, only independent variables are considered; the predicted value is irrelevant (Lakkaraju et al., 2016; Molnar, 2019).

$$cover(r) = \{e_i = (v_{i,1}, v_{i,2}, \dots, v_{i,d}) : i \in \{1, \dots, n\}, e_i \in D, \forall_{j \in \{1, \dots, d\}} v_{i,j} \in t_j\} \quad (3.3)$$

$$Coverage(r) = \frac{|cover(r)|}{n} \quad (3.4)$$

Rule overlap, another metric derived from the rule coverage set, quantifies the extent to which two rules apply to the same examples (Lakkaraju et al., 2016). Formally, for a pair of rules, r_1 and r_2 , rule overlap is the set of examples that satisfy the conditions of both rules, calculated as the intersection of their respective coverage sets (Equation 3.5). Non-zero overlap, where rules share covered examples, require a conflict resolution function for prediction, which can reduce interpretability. Minimizing overlap is therefore preferable, promoting each rule covering a distinct region of the decision space. This highlights an important difference between decision trees,

which are inherently non-overlapping due to their mutually exclusive branching, and classification rule sets, which may exhibit overlap if not explicitly designed to be mutually exclusive.

$$Overlap(r_1, r_2) = cover(r_1) \cap cover(r_2) \quad (3.5)$$

Rule's performance. The predictive performance of a rule can be measured using an approach similar to that used for evaluating a predictive model. The difference is that the evaluation focuses on a single rule rather than all the rules within a model. To formally define rule evaluation metrics, the rule's representation in logical induction is used: $Head \leftarrow Body$ (or $H \leftarrow B$). In this representation, $Head$ represents the class predicted by the rule ($\hat{y}$), and $Body$ represents the itemset that triggers the rule (s). Considering a set of examples from a dataset (either training or testing), the results of applying each example to a given rule r can be organized into a rule confusion matrix table (Table 3.1) (Lavrač et al., 1999; Gomes, 2008).

Table 3.1: Rule confusion matrix

	$e \in cover(r)$	$e \notin cover(r)$	Total
$y = \hat{y}$	HB (TP)	$H\bar{B}$ (FN)	H (P_a)
$y \neq \hat{y}$	$\bar{H}B$ (FP)	$\bar{H}\bar{B}$ (TN)	$\bar{H}$ (N_a)
Total	B (P_p)	$\bar{B}$ (N_p)	n

- B – Represents the number of examples that activate the rule indicating membership in the rule's coverage set, corresponding to predicted positives (P_p);
- $\bar{B}$ – Represents the number of examples that do not activate the rule thereby falling outside the coverage set, corresponding to predicted negatives (N_p);
- H – Represents the number of examples where the predicted class matches the actual class, regardless of whether the rule covers them, corresponding to actual positives (P_a);
- $\bar{H}$ – Represents the number of examples where the predicted class differs from the actual class, regardless of whether the rule covers them, corresponding to actual negatives (N_a).
- HB – Represents the number of correctly predicted positive examples that are covered by the rule, corresponding to true positives (TP);
- $\bar{H}B$ – Represents the number of incorrectly predicted positive examples that are covered by the rule, corresponding to false positives (FP);
- $\bar{H}\bar{B}$ – Represents the number of correctly predicted negative examples that are not covered by the rule, corresponding to true negatives (TN);
- $H\bar{B}$ – Represents the number of incorrectly predicted negative examples that are not covered by the rule, corresponding to false negatives (FN).

Related Work

The cardinality of the *coverage set*, denoted as B , can be derived from the confusion matrix (Lavrač et al., 1999). This allows for an alternative definition of *rule coverage*, distinct from Equation 3.4, as the proportion of examples covered by the rule’s antecedent relative to the entire dataset (Equation 3.6). Thus, while not a performance metric itself, coverage can be obtained from the confusion matrix.

$$\text{Coverage}(r) = \frac{B}{n} \quad (3.6)$$

Rule support represents the fraction of positive examples in the dataset correctly classified by the rule (Equation 3.7). In the context of association rules, support is also known simply as *Frequency* (Lavrač et al., 1999). However, other sources consider coverage and support as synonymous (Molnar, 2019).

$$\text{Support}(r) = \frac{HB}{n} \quad (3.7)$$

Rule accuracy is defined as the fraction of examples correctly classified by the rule among all examples that satisfy the rule’s condition (Equation 3.8). Alternative designations include *Positive reliability* or *Consistency* (Rapp et al., 2019), *Certainty factor* (Quinlan, 1987a), and *Confidence* (especially common in association rule mining) (Lavrač et al., 1999). This variety of terms leads to confusion because, in general model evaluation, the term that aligns with this definition is actually *Precision*. Precision is calculated as $TP/(FP+TP)$ (Lavrač et al., 1999; Gomes, 2008; Molnar, 2019). In contrast, *Accuracy*, in typical model evaluation, refers to the overall fraction of correct predictions across the entire dataset, calculated as $(TP+TN)/(FP+TP+FN+TN)$. Therefore, some authors recommend using the term *Precision* when referring to rule-specific accuracy, thus aligning it with the standard terminology used in model evaluation (Rapp et al., 2019). Furthermore, the term *Hamming accuracy* is sometimes used to specifically denote this rule-specific accuracy (Equation 3.9), further aligning it from the broader concept of accuracy in general model evaluation (Corani et al., 2014).

$$\text{Accuracy}(r) = \frac{HB}{B} \quad (3.8) \quad \text{HAccuracy}(r) = \frac{HB + \overline{HB}}{n} \quad (3.9)$$

The complement of Rule accuracy (or Precision, as discussed above), is *Negative reliability*, defined as the fraction of examples with a class different from the one predicted by the rule that do not satisfy the rule’s condition, relative to the total number of examples not covered by the rule (Equation 3.10). *Rule sensitivity*, also known as *Rule recall*, is the fraction of examples correctly classified by the rule among all examples that truly belong to the class predicted by the rule (Lavrač et al., 1999; Rapp et al., 2019) (Equation 3.11). The complementary metric, *Rule specificity*, is the fraction of examples correctly identified as not belonging to the class predicted by the rule among all examples that truly do not belong to that class (Lavrač et al., 1999), again considering both covered and uncovered examples (Equation 3.12).

$$NegReliability(r) = \frac{\overline{HB}}{\overline{B}} \quad (3.10)$$

$$Sensitivity(r) = \frac{HB}{H} \quad (3.11) \quad Specificity(r) = \frac{\overline{HB}}{\overline{H}} \quad (3.12)$$

The *Rule false discovery rate* measures the proportion of false positives among all examples classified as positive by the rule. This occurs when examples satisfy the rule’s condition but do not belong to the class predicted by the rule. (Equation 3.13). The *Rule weighted false discovery rate* considers the proportion of false positives relative to the total number of examples in the dataset. Formally, the weighted false discovery rate of a rule r , denoted as $WFDR(r)$, is equivalent to multiplying the FDR by the coverage (Equation 3.14).

$$FDR(r) = \frac{\overline{HB}}{B} \quad (3.13) \quad WFDR(r) = FDR(r) \times Coverage(r) = \frac{\overline{HB}}{n} \quad (3.14)$$

3.1.4.2 Functional measures in rule-based models

This section discusses how to assess the interpretability of rule-based models as a whole. A rule-based model (M) can be defined by two components: a set of independent variables (X) and a set of rules ($\mathcal{R}$) (Equation 3.15). Each rule (r) within the model is assumed to be a member of this rule set ($r \in \mathcal{R}, \forall r$). While predictive performance is an important evaluation criteria, this section’s focus lies in model interpretability. The predictive performance of a rule-based model, much like any other predictive model, is evaluated using standard metrics such as Accuracy, Precision, Recall, and F1 derived from the confusion matrix. To evaluate interpretability, this section presents metrics that quantify model complexity, coverage, and fidelity.

$$M = (X, \mathcal{R}) \quad (3.15)$$

Model’s complexity. The *Size* of a rule-based model, measured by the number of rules (R) in its rule set (Equation 3.16), directly relates to its interpretability. Smaller rule sets generally lead to more intuitive and understandable models, exhibiting the property of *simulatability*, which allows a human to grasp the entire model at once (Lipton, 2018). In decision trees, the size corresponds to the number of leaf nodes (Burkart and Huber, 2021).

$$Size(M) = |\mathcal{R}| = R \quad (3.16)$$

Increased complexity in rule-based models directly undermines their interpretability. Complex models, characterized by lengthy rules with numerous conditions, become difficult to comprehend and reason through. While the number of internal nodes are a complexity metric for decision trees (Craven and Shavlik, 1996), a more general measure for rule-based models is the total count of propositions across all rules. This count reflects the cognitive load needed to understand the

model's logic, hindering simulatability. Consequently, a rule-based model's overall *Complexity* is determined by summing the lengths of all individual rules (Equation 3.17).

$$Complexity(M) = \sum_{i=1}^R Length(r_i) \quad (3.17)$$

Average rule length, reflecting the mean number of propositions per rule is another measure of model complexity. For decision sets (e.g., from decision trees), with their unordered, exhaustive, and mutually exclusive rules, the calculation is the model's complexity divided by its rule count. However, for decision lists, the ordered nature of the rules makes the calculation more intricate (Lakkaraju et al., 2016). Since each rule only applies to cases not handled by prior rules, a prediction's effective complexity is the cumulative length of all rules up to and including the triggered one (Equation 3.18).

$$AverageRuleLength(M) = \begin{cases} \frac{Complexity(M)}{R} & \text{if } M \text{ is a decision set} \\ \frac{\sum_{i=1}^R \sum_{j=1}^i Length(r_j)}{R} & \text{if } M \text{ is a decision list} \end{cases} \quad (3.18)$$

Depth (Rüping, 2005) is a complexity metric used for decision trees, defined as the length of the longest path from the root node to any leaf node (Rivest, 1987). Formally, it corresponds to the maximum length among all the rules in the underlying decision set (Equation 3.19).

$$Depth(M) = \max(\{Length(r_i) : 1 \leq i \leq R\}) \quad (3.19)$$

Model sparsity is a multifaceted concept with different interpretations, all of which relate to model complexity. Traditionally, it refers to the number of non-zero coefficients or terms in a model, acting as a direct measure of complexity (Ustun and Rudin, 2016). This aligns with the *sparsity assumption* where fewer terms contribute to a simpler, more interpretable model, similar to how fewer rules or shorter rule lengths reduce complexity, as seen in metrics like Size and Average Rule Length discussed earlier. Alternatively, sparsity can be viewed from a variable-based perspective: a *sparse* model uses only a subset of the available variables. This also increases interpretability and ease of estimation compared to a *dense model* that uses all variables (James et al., 2013; Hastie et al., 2015), similarly to how limiting features simplifies rule-based models. Within decision trees, where variables act as parameters, sparsity often refers to the number of variables used for splitting (Burkart and Huber, 2021; Su et al., 2016). However, some studies define sparsity in decision trees based on the number of leaves (Hu et al., 2019; Lin et al., 2020), aligning with the size concept, leading to a lack of universal agreement on the exact terminology.

Model dimensionality, which corresponds to one perspective on sparsity, is defined as the number of variables used by a model. While the term *dimensionality* is frequently used to describe datasets, it is also a characteristic in a model, particularly concerning its interpretability. High-dimensional models can challenge human cognitive capacity, hindering comprehension of the

logic driving their predictions (Molnar, 2019). Conversely, lower-dimensional models are more transparent, as the relationships between variables and outcomes are more readily traceable (Fretitas, 2014). For rule-based models, representing them as a decision table allows for straightforward determination of their dimensionality, which corresponds to the number of columns, equivalent to the cardinality of the independent variable set X (Equation 3.20).

$$\text{Dimensionality}(M) = |X| \quad (3.20)$$

Model's coverage. Rule coverage evaluates the proportion of training examples covered by a single rule. Averaging this value across all rules in a model provides the average coverage of the entire model. *Fraction covered* (Equation 3.21) of a rule-based model ranges from 1, representing complete coverage of training examples by at least one rule, to 0, indicating a model with no rules and, thus, no coverage (Lakkaraju et al., 2016). It is important to note that this metric can also be used to evaluate a model's completeness when applied to a test set, indicating the percentage of examples in that test set for which the model can provide a prediction.

$$\text{FracCovered}(M) = \frac{1}{n} \left| \bigcup_{i=1}^R \text{cover}(r_i) \right| \quad (3.21)$$

Fraction overlap (Equation 3.22) measures the average overlap between pair of rules within a rule-based model. It ranges from 0, indicating that there is no overlap between any pair of rules, to 1, indicating that all rules cover identical sets of examples (Lakkaraju et al., 2016). Lower values are generally preferred for interpretability, as distinct rules are easier to comprehend than overlapping ones. Different model types exhibit different potential for overlap. In decision sets, overlap is always zero due to the mutually exclusive nature of their rules. In contrast, decision lists are not bound by this constraint, and overlap between rules can occur.

$$\text{FracOverlap}(M) = \frac{2}{R(R-1)} \sum_{\substack{i=1 \\ j=i+1}}^R \frac{|\text{Overlap}(r_i, r_j)|}{n} \quad (3.22)$$

Beyond metrics designed for individual rule-based models, some efforts have focused on quantifying the interpretability of algorithms themselves. One example is the *Interpretability score* (Margot and Luta, 2021). This score assesses an algorithm's interpretability by combining three components: a predictivity score, a stability score, and a simplicity score. The *Predictivity score* evaluates the model's performance relative to a baseline, relying on standard evaluation metrics. The *Stability score* measures the algorithm's sensitivity to variations in training data, with stable algorithms producing consistent results across slightly different datasets. The *Simplicity score* assesses the ease of understanding the model's decision-making process. This component aligns with previously discussed model complexity metrics, such as Size (Equation 3.16), as it considers the number and length of rules. Formally, the Interpretability Score can be represented at a high level as a weighted sum of the three components, where the specific weights (α_1 , α_2 , and α_3) reflect the relative importance assigned to each and sum to 1, for a learning algorithm $\mathcal{L}$

(Equation 3.23). Although the Interpretability Score is useful for comparing algorithms, it does not provide a direct method for evaluating the interpretability of specific models generated by them.

$$\text{InterpretabilityScore}(\mathcal{L}) = (\alpha_1 \times \text{Predictivity}) + (\alpha_2 \times \text{Stability}) + (\alpha_3 \times \text{Simplicity}) \quad (3.23)$$

Model’s fidelity. Fidelity typically arises in post-hoc interpretability, particularly when a simpler white-box model attempts to emulate the behavior of a complex black-box model. A characteristic of a successful surrogate model is that its predictions are faithful to those of the black-box reference (Guidotti et al., 2022). Therefore, fidelity measures how well an interpretable model mimics or accurately imitates the black-box model (Barakat and Diederich, 2005; Guidotti et al., 2018). The literature describes scenarios involving the extraction of Classification rules (Johansson et al., 2004; Zhou, 2004), Decision trees (Confalonieri et al., 2020) from Neural networks (Naveen et al., 2010), or Support vector machines (Farquad et al., 2010).

Accuracy and fidelity are closely intertwined. When a black-box model exhibits high accuracy and the explanation closely mirrors its behavior (high fidelity), the explanation is also highly accurate in reflecting the model’s predictions. Still, some explanations offer only local fidelity, accurately approximating the model’s prediction for specific examples or groups (Ribeiro et al., 2016; Carvalho et al., 2019; Molnar, 2019; Shankaranarayana and Runje, 2019). In certain contexts, however, fidelity is used synonymously with the accuracy of an explanation, reflecting the true correspondence between the explanation and the model’s prediction (Bibal and Frénay, 2016). Without achieving accuracy, an explanation becomes ineffective, failing to faithfully represent the prediction it intends to clarify.

Fidelity is the agreement between the predictions of two models, M_1 and M_2 , on a dataset $D = \{e_1, \dots, e_n\}$ (Figure 3.4). This agreement is represented by a result from a generic evaluation measure denoted by z . Formally, for a classification problem, the *Fidelity Score* (Equation 3.25) is the proportion of examples in D where the predictions from both models, $\hat{y}_i^{(j)}$ for model M_j (Equation 3.24), match (Barakat and Diederich, 2005; Ming et al., 2019). For regression problems, a similar metric can be used, such as the *Root Mean Square Error* (RMSE) (Han et al., 2012) calculated between all corresponding predictions from the two models (Messalas et al., 2019). A value of 1 indicates total agreement, i.e., the predictions of both models for each example are identical. A value of 0 signifies total disagreement, meaning no predictions from the two models match for any dataset example. In either case, the actual value of the target variable is not evaluated, only the predictions.

$$\hat{y}_i^{(j)} = \text{Predict}(M_j, e_i) \quad (3.24)$$

$$\text{FidelityScore}(M_1, M_2, D) = \frac{1}{n} \left| \left\{ e_i : e_i \in D \wedge \hat{y}_i^{(1)} = \hat{y}_i^{(2)} \right\} \right| \quad (3.25)$$

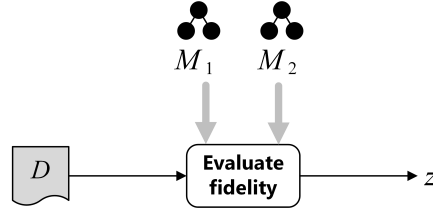


Figure 3.4: Standard approach to fidelity evaluation

In classification problems, traditional fidelity measures the agreement between two models without giving precedence to either. However, when using a white-box model (M_2) to explain a black-box model (M_1), the perspective on fidelity shifts. M_1 becomes the reference model, and its predictions serve as the benchmark for agreement (acting as the *ground truth*). Similar to evaluating predictive performance, where a model's predictions are compared against actual target values, fidelity assessment compares the predictions of M_1 and M_2 . Given this assumption, a confusion matrix, as shown in Table 3.2, can be used to evaluate the agreement between two models. This approach adapts metrics traditionally used for evaluating predictive performance to the task of measuring fidelity (Guidotti et al., 2018).

When comparing the predictions of two models, they can be classified as agreements or disagreements. *Agreements* occur when both models predict the same outcome for a given example: positive agreements (PA) if both predict positive, and negative agreements (NA) if both predict negative. These correspond to TP and TN in traditional performance evaluation. Disagreements occur when the models predict different outcomes: positive disagreements (PD) when M_1 predicts positive and M_2 negative, and negative disagreements (ND) when the reverse is true. These correspond to FP and FN . For each model, the positive predictions can be summed ($P_{M_1} = PA + PD$; $P_{M_2} = PA + ND$) as can the negative predictions ($N_{M_1} = ND + NA$; $N_{M_2} = PD + NA$). These counts are analogous to P_p , P_n , P_a and N_a in a traditional confusion matrix. The total number of examples is the sum of all agreement and disagreement outcomes: $n = PA + PD + ND + NA$.

Fidelity Accuracy (Equation 3.26) directly mirrors the Fidelity Score (Equations 3.25), measuring the degree of alignment between the predictions of the two models. In this context, the metric is denoted as *FidAccuracy* to avoid ambiguity with *Accuracy* in evaluating predictive performance.

Table 3.2: Confusion matrix to evaluate the agreement between two models

		Predictions of M_2		Total
		Positive	Negative	
Predictions of M_1	Positive	PA (TP)	PD (FN)	P_{M_1}
	Negative	ND (FP)	NA (TN)	N_{M_1}
Total		P_{M_2}	N_{M_2}	n

$$FidAccuracy = \frac{PA + NA}{PA + NA + PD + ND} = \frac{\#Agreements}{n} \quad (3.26)$$

Fidelity Precision (Equation 3.27) focuses on the consistency of M_2 's positive classifications with those of M_1 . It measures the proportion of examples where both M_2 and M_1 agree on a positive classification out of all the cases that M_2 predicted to be positive. *Fidelity Recall* (Equation 3.28) shifts the focus to the ability of M_2 to identify all positive examples classified as such by M_1 . It measures the proportion of instances where both M_2 and M_1 agree on a positive classification out of all the cases that M_1 predicted to be positive.

$$FidPrecision = \frac{PA}{PA + ND} \quad (3.27) \quad FidRecall = \frac{PA}{PA + PD} \quad (3.28)$$

Both metrics provide complementary perspectives on the agreement between models M_1 and M_2 . High precision indicates that M_2 's positive classifications are generally consistent with M_1 , while high recall ensures that most positive cases identified by M_1 are also recognized by M_2 . As in predictive performance evaluation, the F1 score, the harmonic mean of precision and recall, can summarize the overall agreement between the two models in a single metric.

As with many concepts in machine learning, the term *fidelity* is applied broadly, encompassing various definitions and applications. In explainable machine learning, fidelity generally refers to how accurately an explanation reflects the underlying model's behavior. This can be further broken down into *external fidelity*, which measures the agreement between a model's predictions and its explanation's implied decisions (Velmurugan et al., 2021), and *internal fidelity*, which assesses the consistency of variable importance rankings between the original and surrogate models (Messalas et al., 2019). Metrics such as *Top_jSimilarity*, often used in conjunction with SHAP values (Lundberg and Lee, 2017), quantify internal fidelity by comparing the rankings of the most influential variables. Within single-model contexts, fidelity has been linked to the correspondence between predictions and the population distribution of independent variables (Yang, 2019), or to the degree to which a variable's assigned importance aligns with its actual contribution to model performance (Flora et al., 2022b). In the emerging field of *Multi-Fidelity Learning*, fidelity has been described as the construction of accurate surrogate models based on data of varying accuracy levels (Pilania et al., 2020).

3.1.5 Interpretability challenges in rule-based models

While the preceding sections have outlined various metrics and approaches for evaluating interpretability, rule-based models, specifically decision trees and classification rules, still face significant interpretability challenges discussed in the following sections.

3.1.5.1 Interpretability of decision trees

From a theoretical standpoint, decision trees have several factors that enhance their inherent interpretability (Lior and Maimon, 2008). Their graphical structure provides a clear visual representation of the decision-making process, assisting in understanding. Decision trees focus on a subset

of relevant variables, simplifying analysis and highlighting those most essential for predictions. Their hierarchical arrangement also implicitly ranks variables, with those closer to the root having more impact on the predictions. Additionally, their flexibility in handling numerical values by allowing a variable to appear multiple times in a single path ensures an accurate representation of thresholds.

While prioritizing attributes with shallower depths is a common decision tree heuristic, it can overlook situations where more profound attributes hold greater importance (Freitas, 2014). Even in simple cases where two variables, x_1 and x_2 , appear only once in the tree, x_2 may classify more examples despite its deeper placement. A more intuitive approach is assessing variable importance based on the number of examples each classifies. This involves summing the examples covered by any leaf node whose path from the root includes that variable. This method allows for variables appearing in multiple paths and gives a clear interpretation of importance based on the dataset coverage of each attribute.

While guided by Occam’s Razor (a preference for simple explanations), decision trees often become overly complex (Domingos, 1999). This complexity arises from the addition of irrelevant branches when a variable is selected for a node. Since all variable values must be included, this can create unnecessary branches even if the values do not contribute to improved predictive performance (Bramer, 2007; Cendrowska, 1987). These irrelevant nodes can mislead users, restrict generalization to new data (Merz, 1998), and lead to overfitting (Ying, 2019), which happens when the model captures minor, often noisy, data variations instead of broader patterns. This complexity also directly impacts interpretability, particularly in deeper trees (Freitas, 2014; Guidotti et al., 2018).

To prevent overfitting, decision tree algorithms use methods to simplify into pruned rule sets (Quinlan, 1987b). Pruning involves replacing some subtrees with leaves (Almuallim, 1996). This *trimming* process ensures a balance between simplicity and accuracy, enabling the model to capture essential trends without being overwhelmed by irrelevant details. The primary motivation behind pruning is to introduce a bias towards simpler models, thus avoiding complex rules with low coverage that may contain irrelevant tests added only to exclude noisy examples (Fürnkranz, 1997). This issue persists even in noise-free scenarios arising from the intrinsic structure of decision trees.

3.1.5.2 Interpretability of classification rules

Despite their shared role as rule-based models, decision trees and classification rules offer differing levels of interpretability due to their presentation formats. Decision trees exploit a hierarchical structure where variable position directly reflects importance, with those closer to the root holding more significant influence. However, this inherent advantage is absent in classification rules due to their flat, textual format. Attempts have been made to bridge this gap by arranging rule conditions and variables based on estimated relevance (Freitas, 2014). However, capturing the overall importance of a variable across all rules remains challenging. A common approach involves summing

the number of examples classified by rules containing the variable, but this method provides a quantitative yet potentially limited measure.

While decision trees offer a holistic and simultaneous view of the model through their visual structure (Lipton, 2018), their textual counterparts, classification rules, can be analyzed as individual, self-contained knowledge units, as local patterns (Guidotti et al., 2018). This perspective allows users to focus on specific elements, like examining individual tree paths. However, some argue that converting a decision tree into separate rules further improves comprehensibility (Quinlan, 1993). This creates a trade-off: understanding the model’s overall logic through its structure versus gaining granular insights from individual rules. While some consider decision lists simpler (Weiss and Indurkha, 1993), this trade-off highlights the different aspects of interpretability offered by these representations.

Pruning strategies differ based on the model format. Any condition can be removed freely in rule sets since each rule function is an independent unit unbound by a hierarchical structure. In contrast, decision trees prioritize removing deeper nodes further down the tree over shallower ones closer to the root. This preference arises due to the inherent hierarchy of trees, where shallower attributes establish foundational decisions that influence subsequent ones (Freitas, 2014).

While the total number of conditions across all rules is a standard complexity metric (Equation 3.17), it can be misleading. This is because it ignores how different examples may require varying numbers of rules for classification due to the ordered structure (Freitas, 2014). To address this, a more accurate measure is the average number of rule conditions evaluated per example on a test set (Otero and Freitas, 2013). This directly captures the *prediction-explanation size* of the decision list as the complexity users encounter when understanding the classification process for individual examples.

3.2 Methods to merge prediction models

This section focuses on various approaches to merging prediction models. It is important to distinguish these approaches from ensemble learning (often referred to as *model combination*). While some ensemble methods involve creating multiple models independently and subsequently combining their predictions for classification or regression tasks (e.g., bagging, random forests), other methods, such as boosting (e.g., AdaBoost) and Negative Correlation Learning, build models sequentially, with each model’s training influenced by the performance of its predecessors. In contrast to all ensemble methods, model merging focuses on directly combining the models themselves by integrating components from various models to construct a new model.

3.2.1 Combining models: ensemble vs. merging methods

Ensemble learning (Opitz and Maclin, 1999) has become a prominent technique in machine learning. Its main objective is to improve the accuracy and reliability of predictions, making it particularly effective in situations where individual models may exhibit weaknesses or biases (Garcia-Pedrajas et al., 2005). Ensemble learning combines a set of models, each created by applying a

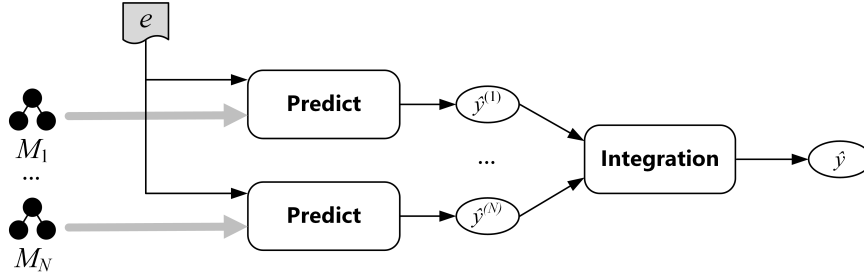


Figure 3.5: Ensemble method setup

learning process to a given problem. The models in the ensemble are integrated using a particular method to produce the final prediction (Mendes-Moreira et al., 2012). Ensemble methods are typically approached through one of two mechanisms: either the predictions of the base models are combined to yield an ensemble prediction (combination/fusion approach), or the prediction of one base model is selected according to some criteria to form the final prediction (selection approach) (Rooney et al., 2004; Kuncheva, 2002). However, selection can be treated as a particular case of combination where only one model receives a non-zero weight (Mendes-Moreira et al., 2012). Figure 3.5 depicts the setup of an ensemble method, where an example, if submitted to a set (or ensemble) of N models M_1 through M_N , each providing a prediction, $\hat{y}_1$ through $\hat{y}_N$ respectively. The integration stage subsequently processes these individual predictions to generate the final ensemble prediction ($\hat{y}$).

There are a number of popular ensemble methods in machine learning. *Bootstrap Aggregating*, also known as *Bagging* (Breiman, 1996), addresses overfitting and enhances model stability by training multiple copies of the same base model on different subsets of the training data. This helps to diversify the models and avoid overreliance on specific training examples. *Boosting* is a family of algorithms that train models sequentially, with each subsequent model focusing on improving the errors made by the previous ones. *AdaBoost* (Freund and Schapire, 1999) (Adaptive Boosting) is a specific and popular boosting algorithm that iteratively trains weak learners and assigns varying weights to training examples based on their previous classification performance. Examples that previous models misclassified receive higher weights, focusing the training process on these challenging cases. *Random Forests* (Breiman, 2001) build upon Bagging by training multiple decision trees on different subsets of the training data and also using random subsets of features at each split. This added randomness further diversifies the trees and helps to avoid overfitting. *Gradient Boosting* is another boosting technique that builds models sequentially, where each new model minimizes a loss function by using the gradient descent algorithm. *XGBoost* (Chen and Guestrin, 2016) (Extreme Gradient Boosting) is an optimized and highly efficient implementation of gradient boosting that incorporates several enhancements, such as regularization and parallel processing, to handle large datasets and achieve state-of-the-art performance.

Ensemble methods are a specific type of metamodel. *Metamodels*, or higher-order models, are supervised learning approaches that combine simpler base models to enhance predictive performance (Polikar, 2006; Rokach, 2010). To illustrate, there are three main methods for combining

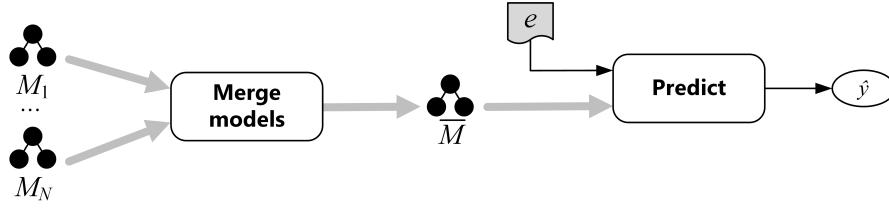


Figure 3.6: Merging method setup

Neural networks (Freitas and Rodrigues, 2006): *Model mixing* (combining estimates), *Model synthesis* (merging model structures), and *Model switching* (selecting an estimate). While ensemble learning aligns with model mixing and switching, model synthesis differs by directly merging model components. Though initially proposed for neural networks, the idea of model synthesis is particularly relevant for rule-based models due to their modularity and ease of combining individual rules.

Model merging formally involves blending N individual models (M_1 through M_N) into a single model (denoted as $\bar{M}$), as depicted in Figure 3.6. As discussed in Section 1.2, the individual models can act as operational models and the merged model as a consensus model. Even though both methods use multiple models, their goals and outcomes differ significantly. Ensemble learning focuses on improving prediction accuracy, while model merging aims to combine models while preserving interpretability.

The terms *merging models* and *combining models* are sometimes used interchangeably in the literature (Williams, 1988; Hall et al., 1998; Shannon and Banks, 1999; Obregon et al., 2019). However, the term *combination* increasingly refers to ensemble methods in recent research (Weiss and Indurkha, 1995; Merz, 1998; Gama, 1999; Kuncheva, 2002; Tong et al., 2003). For clarity, this document distinguishes between the two terms often associated with combining models:

- Ensemble method – A process to combine the predictions from multiple models without merging their internal components;
- Merging method – A process to combine the components of multiple models into a single model that makes predictions.

Merging models have been addressed through various methods, broadly categorized into two main approaches. The first approach is analytical, which involves identifying and extracting rules from a set of models. These rules are then combined and reorganized to create a new model. The second approach is mathematical, where a mathematical function is applied to a set of models. This function manipulates the models and generates a new model as a result. The following sections discuss analytical and mathematical approaches in further detail.

3.2.2 Analytical approaches to merge models

Analytical approaches were first introduced in the late eighties (Williams, 1988) and then branched out as distinct variations by several researchers. The motivation has been to address two kinds

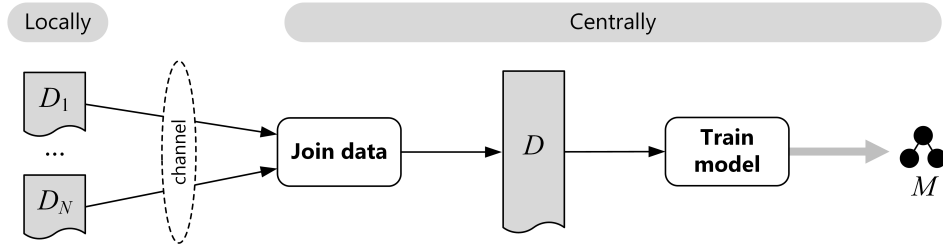


Figure 3.7: Creating a model by moving datasets

of problems. The first was to generate models for distributed systems, i.e., wherever the data was dispersed throughout numerous locations, described as “mining data distributed on distant machines” and “connected by low transparency connections” (Bursteinas and Long, 2001) and the second resulted from the expansion in the volume of data gathered by information systems. This reality prompted the need to generate models that could handle large datasets (Maimon and Rokach, 2010). There was a noticeable scarcity of resources to carry out the task, described as a prolonged learning process sometimes overwhelming the system memory (Hall et al., 1998) and the emergence of datasets exceeding available memory (Andrzejak et al., 2013). Both motivations are detailed in the following sections.

3.2.2.1 Merging models to create a model from distributed data

One of the first contexts where the problem of merging prediction models emerged was to create models for systems in naturally distributed data environments, i.e., where the data sources are spread over different locations (Bursteinas and Long, 2001), with dataset having identical format and structure ($D_1, \dots, D_N$). All local datasets are sent to a single location and integrated into one dataset, often named a *monolithic dataset* (D). A single model M is then created centrally using all data, as depicted in Figure 3.7. This approach, however, introduces difficulties as moving data may not be safe; it may be expensive or plain unattainable due to its large volume.

Alternatively, the models can be moved instead of moving data, as presented in Figure 3.8. Models are created in each location ($M_1, \dots, M_N$), moved to a centralized location, and merged into a single model $\bar{M}$ (Provost and Hennessy, 1996; Bursteinas and Long, 2001).

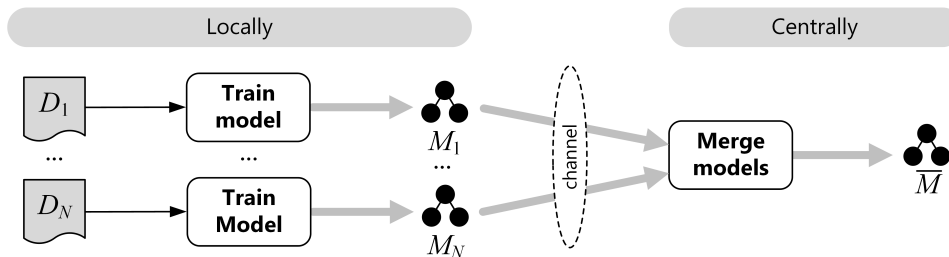


Figure 3.8: Creating a new model by moving and merging models

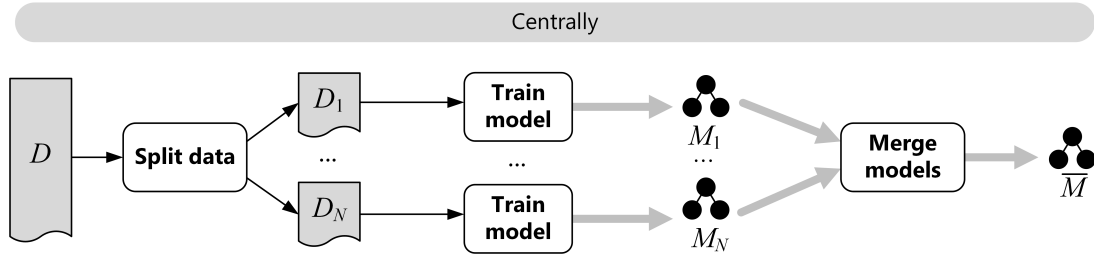


Figure 3.9: Merging models to create a model from a large dataset

The merging process has been showcased in approaches to merging a pair of decision trees, each addressing specific issues. One approach divides the problem into two scenarios: one for disjoint datasets, where each example belongs only to one dataset, and another for overlapping datasets, where examples are shared between datasets (Bursteinas and Long, 2001). In the former scenario, the models are merged without any modifications (by placing the new models as leaves of a meta-model), while in the latter, the models are intersected. Other alternatives include different ways of carrying out the process, namely, *tree-merging one-pass*, *tree-merging multiple-pass*, and *voting* (Andrzejak et al., 2013). The last of the three has the disadvantage of not being interpretable. The process quality is evaluated by merging pairs of decision trees trained using datasets from the UCI Machine Learning Repository (Markelle Kelly et al., 2023). In both scenarios, the results compare the accuracy of the merged model with that of a model induced on a monolithic dataset. However, overall, the results are not consistent, with some studies finding the accuracy to be similar to the model induced on the monolithic dataset (Bursteinas and Long, 2001), while others argue that it is generally worse (Andrzejak et al., 2013).

3.2.2.2 Merging models to create a model from a large dataset

An additional context for merging models has arisen due to the growing volume of data information systems collect, requiring methods to create models from large datasets (Maimon and Rokach, 2010). To tackle this problem, developing an artificial distributed data environment has become critical. This setup involves dividing a large dataset into N datasets, as many as necessary until it becomes feasible to create a model for each (Andrzejak et al., 2013). Subsequently, all operational models are merged into one ($\bar{M}$), as depicted in Figure 3.9.

Evaluating merged rule-based models requires striving for both performance and rule equivalence. While the initial aim was to replicate the accuracy of a single model trained on all data (Provost and Hennessy, 1996), studies point to potential accuracy drops (Chan and Stolfo, 1997). Therefore, the ideal merged model should be *as close as possible* to its monolithic counterpart. This means maintaining similar accuracy while capturing equivalent decision-making logic through its rules, even if the structure differs. Achieving both is essential for effective merging.

Not all merging techniques create individual models for each data subset. Some generate rules in each subset but only include those meeting a specific performance threshold in the final model (Provost and Hennessy, 1996). This implies training rules on one dataset and evaluating

them on all others. Rules considered *satisfactory* on the entire dataset are kept, as they are considered more generalizable. Another approach partitions the data into disjoint parts and trains independent decision trees on each (Hall et al., 1998). These *individual learners* contribute their knowledge to a combined knowledge base. Initial experiments on two datasets showed promise for this distributed learning approach, with merged models constructed similar to traditional merging methods (Bursteinas and Long, 2001; Andrzejak et al., 2013).

3.2.3 Mathematical approaches to merge models

Mathematical approaches are quite different and have been designed to solve specific problems. While some change the format of the model to allow the application of a mathematical function, others keep the original format and design metrics to represent each model. A mathematical function is applied to the metric, and then one of the models represents the set.

In analyzing time-critical data streams using mobile devices, one approach is based on a method to add decision trees (Kargupta and Byung-Hoon Park, 2004). The method uses the Fourier Transform to decompose a function of time into its frequencies, resulting in its frequency domain representation, also known as its spectrum. The application of this method is to mine critical data streams in online learning, which produces several decision trees. These trees may need to be compared with each other and, if necessary, aggregated. The problem is formulated as the transmission of the models over a wireless network. Since decision trees are functions, they can be defined by a spectrum. Merging models is achieved by adding their spectra. If required, the Inverse Fourier Transform can transform the final model back to the decision tree domain. While this approach has shown promise, it has faced some criticism. One point of contention is its limited scope, as it only considers binary attributes. Additionally, there has been a lack of empirical evidence to support its effectiveness (Andrzejak et al., 2013).

Another approach seeks to merge a set of decision trees by finding a *central tree* (Shannon and Banks, 1999) in a Maximum Likelihood Estimate (MLE) approach. The method is based on a probability distribution on the set of models and a distance metric of structural differences between trees. The latter uses weights to penalize tree differences occurring near the root differentially compared with less serious ones occurring near the terminal nodes. The approach was applied to a set of decision trees from biomedical data with 13 models created to predict the presence or absence of cancer based on immune system parameters. Results show that the merged model retains the interpretability of a single decision tree model.

Another example of a mathematical approach to merging models consists of constructing a tree that is the *nearest* on average to a given set of trees, also referred to as a *supertree* (Gorbunov and Lyubetsky, 2011). The notion of closeness is defined as a function that counts the number of events that occur when comparing each tree to the average one. Examples of such events include divergence, duplication, loss, and transfer. The method is tested in analyzing the evolutionary trees of different species. In this context, the problem involves mapping a set of gene trees into a species tree (the average tree). However, its applicability extends beyond this specific domain. Any scenario where trees and specific operations can represent entities that can be mapped to the four

defined event types is eligible for this method. Recognizing the limitation of non-binary trees in this context, a system for their conversion into binary formats was introduced, thereby expanding the algorithm's potential use cases (Gorbunov and Lyubetsky, 2012). The practical effectiveness of the supertree approach is further demonstrated within the field of molecular biology research, highlighting its potential to impact various domains and showcasing the innovative potential of model merging through strategic mathematical techniques.

3.3 Summary

This chapter focuses on two paramount research areas central to this thesis. The first is interpretability, specifically how it can be defined, classified, and, most importantly, measured, as it will be an objective analysis of consensus models. This is presented as a taxonomy of interpretability techniques and evaluation approaches. The inherent trade-offs between model predictive performance and interpretability are also addressed, recognizing the diverse needs of different stakeholders for clear and meaningful model explanations.

The second research area investigates methodologies employed for merging prediction models. This process is framed within the broader landscape of model combination techniques, drawing comparisons and contrasts with related fields such as ensemble learning. The chapter further distinguishes analytical and mathematical approaches to model merging, highlighting their strengths and limitations.

Related Work

Chapter 4

Inmplode: A Framework to Interpret Multiple Related Rule-based Models

This chapter introduces *Inmplode*, a novel framework for interpreting multiple related rule-based models by constructing consensus models from context-specific operational models. It begins with an overview of the framework, distinguishing the generic workflow from the customizable components, followed by a detailed description of each component. A case study from the University of Porto demonstrates the practical application of the framework through concrete examples. Given the level of detail involved, the core process of merging individual models into a consensus model, central to *Inmplode*, is discussed in greater depth in Chapter 5, where it is addressed as a distinct and standalone problem.

4.1 Model merging in decentralized organizations

Organizations often face complex data analysis problems involving many sources of data. To tackle these challenges, they are adopting a strategy of breaking down these significant problems into smaller, more manageable sub-problems that can be analyzed separately. Rather than relying on a single model to tackle the entire scope of the problem, organizations can use a set of models, each tailored to address a specific sub-problem. These sub-problem-specific models are designed to provide more accurate insights within their particular domains. This approach benefits organizations with dispersed or decentralized activities, as detailed in Section 1.1.2. For example, companies with multiple store locations or universities with various academic programs. In the example of chain stores, each store might have its dedicated model to predict sales performance. Similarly, in the case of universities, individual models might be created to predict student dropout for different programs. This level of granularity helps organizations make more informed decisions based on the intricacies of each business unit.

When organizations need to understand phenomena across multiple business units, the large number of individual models makes it impractical to analyze each one separately. To address this challenge, the proposed approach involves merging the operational models into a single consensus

model. For example, when the operational models are decision trees, this process involves analyzing the rules of each tree and deriving a new decision tree—the consensus model—that effectively summarizes the knowledge contained in the original models.

Inmplode is a framework specifically designed to address the process of training, evaluating, and merging rule-based models into consensus models. Frameworks are essential in machine learning, streamlining the development and application of complex algorithms. They provide foundational platforms with pre-built libraries of methods and structures, simplifying data preprocessing, model construction, and optimization (Pedregosa et al., 2011). Essentially, frameworks offer a high-level abstraction, allowing researchers and practitioners to focus on the core problem rather than the intricacies of specific algorithms.

The need for a generalized framework arises from limitations found in existing methodologies. While addressing seemingly similar problems, these approaches are often narrowly focused and intertwined with specific case studies, hindering their broader applicability. Despite variations in terminology, the core subproblems and concepts remain consistent across these methods. However, their high degree of specialization limits their adaptability to different problem domains. *Inmplode*'s generality originates from accommodating diverse instantiations with a consistent structure and terminology. This fosters a broader understanding of the problem space and enables widely applicable solutions. A key feature is the use of second-order methods, which take other methods as input. This allows for dynamic composition of subproblems and flexible solution pipelines, leading to greater adaptability.

4.2 The *Inmplode* framework

The *Inmplode* framework introduces a novel analytical approach for merging rule-based models, with a modular design that facilitates both flexibility and adaptability. This is achieved through a generalized workflow that can be extended with custom-designed components. This architecture enables the application of *Inmplode*'s underlying principles to various research domains, with the potential for adaptation to specific problem requirements through the development of specialized methods. *Inmplode* revolves around the two fundamental concepts of operational models and consensus models. These models are also the outputs generated by the framework. Therefore, the process aims to achieve two main objectives:

1. Develop rule-based operational models that capture knowledge within a local scope with the following requirements:
 - Predictive performance– Each operational model should have high predictive performance within its designated domain;
 - Interpretability – Operational models should be interpretable, allowing stakeholders to understand the rationale behind their predictions.
2. Develop rule-based consensus models that summarize the knowledge of a set of operational models with the following requirements:

- Fidelity – Consensus models are expected to exhibit some agreement with the predictions of the operational models they represent, although their suitability as surrogates may be limited;
- Interpretability – Consensus models should strive to maintain interpretability, retaining the same format as the underlying operational models;
- Completeness – Consensus models should be able to provide predictions covering the same decision space defined by the operational models.

The specified requirements delineate complementary functions for the two model types for a given phenomenon: operational models prioritize localized predictive performance whereas consensus models seek to synthesize a holistic, though potentially less granular, understanding of the global phenomenon.

4.2.1 Generic workflow

Given N datasets, each with data collected from a distinct business unit, the *Inmplode* framework is a pipeline with a generic workflow, depicted in Figure 4.1. The constituent subprocesses of this workflow are detailed as follows:

- Create folds – Divide each dataset (D_i) into multiple folds to be used for training and evaluating both the operational and consensus models;
- Train operational models – Train an operational model (M_i) for each dataset;
- Evaluate operational models – Perform a multi-criteria evaluation of each operational model (M_i), with the results in set Z_i ;
- Stratify operational models – Group the operational models into J strata (S_j) based on a predefined similarity criterion, resulting in L_j models within each stratum;
- Merge operational models – Within each stratum (S_j), merge the L_j operational models to derive a consensus model ($\bar{M}_j$);
- Evaluate consensus models – Perform a multi-criteria evaluation of each consensus model ($\bar{M}_j$), with the results in set $\bar{Z}_j$.

Data preprocessing is a fundamental step for building any predictive model. However, *Inmplode*'s focus lies on the training and evaluation stages of the process. Consequently, the framework assumes all data preprocessing steps have been completed, and the datasets are prepared for immediate use. The use of rule-based models eases this assumption, as they can often handle raw or minimally processed data, particularly when dealing with categorical features (Hastie et al., 2017).

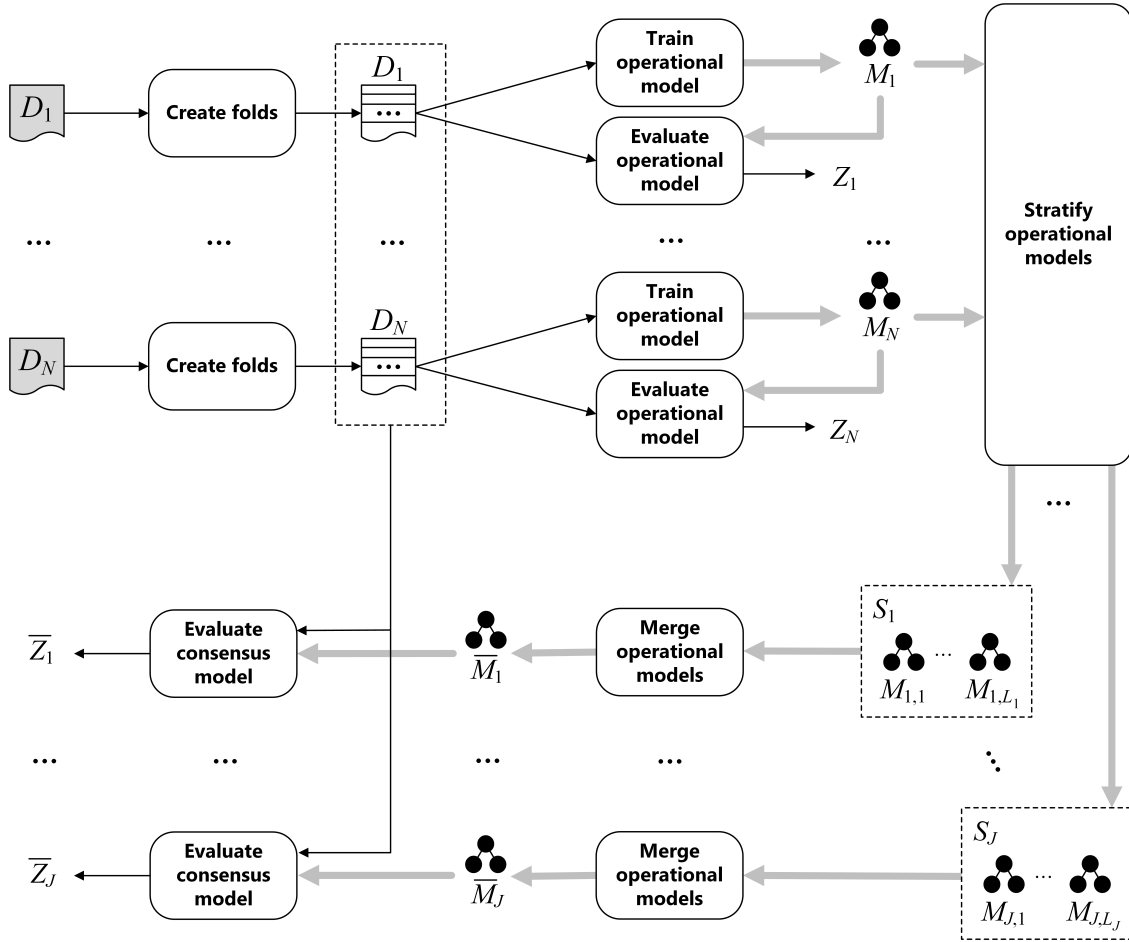


Figure 4.1: The *Inmplode* framework generic workflow

4.2.2 Customizable components

Customizable components are pivotal to the flexibility of the *Inmplode* framework, allowing for fine-grained control over their behavior and customization to specific needs through the specification of input parameters. These components feature a modular design, enabling integration into the framework's workflow and interoperability with other components. Furthermore, their configuration is input-driven, providing a flexible and intuitive mechanism to adjust the components' behavior according to specific requirements.

The *Inmplode* framework uses the following inputs:

- Datasets ($\{D_1, \dots, D_N\}$) – A collection of N datasets, each holding the data relative to a phenomenon or outcome of a process that occurs concurrently. Thus, the examples of each dataset are related to a specific location or a distinct business unit. The datasets all have the same format (the same variables) but may vary in the number of examples.
- Training, evaluation, and stratification – Parameters that customize the creation, evaluation, and grouping of the operational models:

- Independent variables (X) – The set of features used to train the operational models;
 - Target variable (y) – The variable the operational models are trained to predict;
 - Positive class ($\oplus$) – The class designated as positive for model evaluation;
 - Stratification variables ($\mathcal{V}$) – The set of variables used to define the criteria for grouping operational models;
 - Folds (K) – The number of folds used for cross-validation;
 - Learning algorithm ($\mathcal{L}$) – The algorithm used to train rule-based models.
- Evaluation metrics specifications – Two sets of specifications relating to evaluation tasks for:
 - Model evaluation specifications ($\mathcal{Z}$) – A set that describes the criteria and experimental setups to evaluate both operational and consensus models;
 - Rule evaluation metrics (Φ_r) – A set of metrics to evaluate individual rules.
 - Sets of abstract methods – Two sets of abstract methods for implementing customizable tasks related to:
 - Conversion methods ($\mathcal{F}_C$) – To handle models in various formats;
 - Merging methods ($\mathcal{F}_M$) – To develop consensus models with different approaches.

After execution, the *Inmplode* framework organizes its outputs into the following categories:

- Operational models:
 - Strata of operational models ($\mathcal{S}$) – Sets of operational models for deployment arranged into a set of multiple strata according to the criterion specified in the stratification variables ($\mathcal{V}$);
 - Evaluation results (Z_S) – A set with the results of the evaluation of the operational models organized by strata.
- Consensus models:
 - Set of consensus models ($\overline{\mathcal{M}}$) – A set encompassing consensus models created for each stratum;
 - Evaluation results ($Z_{\overline{\mathcal{M}}}$) – A set with the results of the evaluation of the consensus models organized by strata.

4.2.3 Formal overview

Algorithm 1 provides an overview of the *Inmplode* framework, and it inevitably omits many details, which will be explored in the following sections. A case study accompanies this breakdown to illustrate the process and its outcomes. A description of the steps is described next for reference.

As a preliminary step, each dataset ($D_1, \dots, D_N$) is partitioned into K folds using *stratified sampling* by the CreateFolds function. This means the distribution of class labels within each fold reflects the overall distribution in the dataset. Consequently, to each dataset D_i corresponds a set of folds denoted by $\{f_{i,1}, \dots, f_{i,K}\}$. These folds serve a dual purpose:

- Model training – $(K - 1)$ folds are used as the training set for operational models;
- Model evaluation – The remaining fold is reserved for evaluating operational and consensus models.

Rather than selecting a specific fold for operational and consensus model creation and evaluation, *Inmplode* uses k -fold cross-validation by repeating the entire process K times, each repetition is referred to as a *setup*. In setup #1, all folds numbered 1 (denoted as $f_{i,1}$ for dataset D_i) are used as the test set, while the remaining folds from each dataset are combined for training. Similarly, for each subsequent setup (setup #2, setup #3, etc.), the number of the setup is the designated fold as the test set for evaluation. In contrast, the remaining folds are used for training.

By repeating the process with each fold as the test set, *Inmplode* avoids biasing the evaluation results that could occur if a single arbitrary set of data were chosen for testing. Consequently, this approach creates K models (both operational and consensus). As is typical in k -fold cross-validation, the final evaluation results are obtained by averaging the results from each setup.

The primary output of *Inmplode*'s framework is operational and consensus models to be used for deployment. While the K models created during the k -fold cross-validation process serve evaluation purposes, the deployment versions use all available examples from the datasets without considering the fold partitions. These deployment-ready model correspond to those generated by setup #0. Consequently, the process is repeated $K + 1$ times, with evaluations performed only for model versions from setup #1 to set up # K .

Next, the process iterates from setup #0 to setup # K , encompassing three major tasks:

1. Train and evaluate N operational models – An operational model is trained and evaluated for each dataset. The training set for each operational model varies depending on the setup. In setup #0, the training set consists of examples in the entire dataset, while in setups #1 to # K , the training set excludes examples in the fold corresponding to the current setup. Each operational model is trained by the TrainModel function, which uses the specified learning algorithm ($\mathcal{L}$), the training set (Tr), the set of independent variables (X), and the target variable (y). The resulting operational model is denoted as $M_i^{(k)}$, where i refers to the dataset (D_i), and k refers to the setup. Once trained, each operational model is evaluated using the EvaluateModel function. This evaluation is performed according to the model evaluation specifications ($\mathcal{Z}$). It encompasses various criteria of evaluation, using the fold associated with the current setup ($f_{i,k}$) as test data and the specified positive class ($\oplus$), for the metrics that require it. The evaluation results for each criterion are denoted by $Z_i^{(k)}$, following the same notation as the corresponding operational model.

2. Stratify the operational models into J strata – The operational models are stratified into J strata based on the set of stratification variables ($\mathcal{V}$). Each stratum of models can be based on two types of variables. The first is dataset variables (shared among all datasets, resulting in identical strata compositions across setups), and the second is model characteristics (can vary between versions of the same model, leading to setup-specific strata compositions). The result is a set denoted by $\mathcal{S}^{(k)}$, containing the strata. Each stratum contains a set of operational models with similar characteristics, denoted by $\mathcal{M}$.
3. Create and evaluate J consensus models – A consensus model ($\overline{\mathcal{M}}_j^{(k)}$) is created by the MergeModels function requiring the set of models in a specific strata ($\mathcal{M}_j^{(k)}$), the rule evaluation metrics (Φ_r), and the two sets of abstract methods ($\mathcal{F}_C$ and $\mathcal{F}_M$). Similarly to operational models, the subsequent task is evaluation carried out by the EvaluateConsensusModel function, which requires the consensus model, the set of models used to create it, the model evaluation specifications ($\mathcal{Z}$), a set of test sets ($\mathcal{T}$) and the specified positive class ($\oplus$). Unlike the evaluation of operational models, the evaluation of consensus models requires a set of test sets instead of a single test set (as the folds of each of the L models within each stratum need to be specified separately).

After the process is repeated for all setups, the final step is to collect the deployment models and the overall evaluation results. The first (denoted by $\mathcal{S}$) is the sets of operational models grouped by strata for deployment. This set refers to the versions of the operational model sets of setup #0, i.e., those trained using all available data in each dataset. The second (denoted by $Z_{\mathcal{S}}$) is the set with the same structure as the results of the aggregation of the evaluations of each operational model across all setups in each stratum. The third (denoted by $\overline{\mathcal{M}}$) is the set of consensus models according to each stratum for deployment. The fourth (denoted by $Z_{\overline{\mathcal{M}}}$) is the set of the results of the aggregation of the evaluations of each consensus model across all setups in each stratum.

Case study: Predicting student performance at the University of Porto

A real-world case study at the University of Porto, Portugal, served as a testing ground for the *Inmplode* framework. The study centered on student enrollment data for all courses offered during the 2012/2013 academic year. Each example represented a student's enrollment in a specific course, characterized by variables detailed in Table 4.1, also specifying the variable type. While future studies would incorporate a broader range of variables, this initial selection allowed for a practical demonstration of the framework's application in a specific, real-world problem.

Given the focus on predicting student success or failure in each course, the target variable was designated as 'Approval', with the positive class was assigned the value 'no', reflecting the binary nature of the classification task (students either pass or fail). Student enrollment data was partitioned based on the corresponding course to foster the application of the *In-*

Algorithm 1 Inmplode: A framework for training, merging, and evaluating rule-based models

Input:

- Datasets: $\{D_1, \dots, D_N\}$
- Training, evaluation and stratification specifications: $X, y, \oplus, \mathcal{V}, K, \mathcal{L}$
- Evaluation metrics specifications: $\mathcal{Z}, \Phi_r$
- Sets of abstract methods: $\mathcal{F}_C, \mathcal{F}_M$

Output:

- Strata of operational models and evaluation results: $\mathcal{S}, Z_{\mathcal{S}}$
- Consensus models and evaluation results: $\overline{\mathcal{M}}, Z_{\overline{\mathcal{M}}}$

▷ Create K folds for each dataset

for i such that $1 \leq i \leq N$ **do** ▷ N is the number of datasets

$\{f_{i,1}, \dots, f_{i,K}\} \leftarrow \text{CreateFolds}(D_i, K, y)$

end for

for k such that $0 \leq k \leq K$ **do** ▷ Iterate over each setup

▷ Train and evaluate N operational models

for i such that $1 \leq i \leq N$ **do** ▷ Iterate over each dataset

$Tr \leftarrow \begin{cases} D_i \setminus f_{i,k} & k > 0 \text{ } \triangleright \text{ Training set for evaluation model} \\ D_i & k = 0 \text{ } \triangleright \text{ Training set for deployment model} \end{cases}$

$M_i^{(k)} \leftarrow \text{TrainModel}(\mathcal{L}, Tr, X, y)$

$Z_i^{(k)} \leftarrow \text{EvaluateOperationalModel}(M_i^{(k)}, \mathcal{Z}, f_{i,k}, \oplus)$

end for

▷ Stratify the operational models into J strata

$\mathcal{S}^{(k)} \leftarrow \text{StratifyModels}(\{M_1^{(k)}, \dots, M_N^{(k)}\}, \mathcal{V})$

▷ Create and evaluate J consensus models

for j such that $1 \leq j \leq J$ **do** ▷ Iterate over each stratum

$\overline{M}_j^{(k)} \leftarrow \text{MergeModels}(\mathcal{M}_j^{(k)}, \Phi_r, \mathcal{F}_C, \mathcal{F}_M) \triangleright \text{Merge operational models in a stratum}$

$\mathcal{T} \leftarrow \{f_{1,k}, \dots, f_{L_j,k}\} \triangleright L_j \text{ is the number of operational models in stratum } j$

$\overline{Z}_j^{(k)} \leftarrow \text{EvaluateConsensusModel}(\overline{M}_j^{(k)}, \mathcal{M}_j^{(k)}, \{Z_{j,1}^{(k)}, \dots, Z_{j,L_j}^{(k)}\}, \mathcal{Z}, \mathcal{T}, \oplus)$

end for

end for

▷ Collect results

$\mathcal{S} \leftarrow \mathcal{S}^{(0)} \triangleright \text{Sets of operational models grouped by strata for deployment}$

$Z_{\mathcal{S}} \leftarrow \text{EvaluationResultsOperational}(\left\{ \left\{ \left\{ Z_{j,i}^{(k)} : 1 \leq k \leq K \right\} : 1 \leq i \leq L_j \right\} : 1 \leq j \leq J \right\})$

$\overline{\mathcal{M}} \leftarrow \{\overline{M}_1^{(0)}, \dots, \overline{M}_J^{(0)}\} \triangleright \text{Set of consensus models for deployment}$

$Z_{\overline{\mathcal{M}}} \leftarrow \text{EvaluationResultsConsensus}(\left\{ \left\{ \overline{Z}_j^{(k)} : 1 \leq k \leq K \right\} : 1 \leq j \leq J \right\})$

mplode framework. This resulted in 730 individual datasets, each representing a specific course. These courses originated from 391 academic programs offered by the University. While all datasets share the same variable structure, they have significant size variations. Some datasets, particularly those related to Ph.D. programs, might have only one student enrolled. Conversely, multi-program courses could have a maximum of 950 students enrolled.

Table 4.1: Datasets variables in the case study

Variable	Description	Type
$x_1 = \text{Age}$	Age of student at enrollment date	Numerical discrete
$x_2 = \text{Gender}$	Male, Female	Categorical
$x_3 = \text{Marital status}$	Single, Married, Divorced, ...	Categorical
$x_4 = \text{Nationality}$	First nationality of student	Categorical
$x_5 = \text{Displaced}$	The student lives in another district	Categorical
$x_6 = \text{Scholarship}$	The student has a scholarship	Categorical
$x_7 = \text{Special needs}$	The student has disabilities	Categorical
$x_8 = \text{Type of admission}$	Type of application in program	Categorical
$x_9 = \text{Type of student}$	Regular, mobility, extraordinary	Categorical
$x_{10} = \text{Type of dedication}$	Full-time, Part-time	Categorical
$x_{11} = \text{Debt situation}$	There are fees due	Categorical
$x_{12} = \text{Status of student}$	Ordinary, Employed, Athlete, ...	Categorical
$x_{13} = \text{Years of enrollment}$	# of academic years enrolled	Numerical discrete
$x_{14} = \text{Delayed courses}$	# of uncompleted courses	Numerical discrete
$y = \text{Approval}$	The student has passed the course	Categorical

4.3 Creation of folds for each dataset

The *Inmplode* framework uses the k -fold cross-validation (Stone, 1974) method to evaluate the performance of its models. This technique involves dividing each dataset (D_i), representing data collected from a process within a business unit, into a set of K partitions or folds. Each fold is created by randomly sampling a subset of examples from the corresponding dataset (D_i). The goal is for each fold to have approximately the same number of examples (Equation 4.1). The union of all folds for a dataset is equivalent to the original dataset itself (Equation 4.2). The folds are mutually exclusive, meaning each example belongs to only one fold.

$$f = \{e_1, \dots, e_{|f|}\} \quad (4.1) \quad D_i = f_{i,1} \cup \dots \cup f_{i,K} \quad (4.2)$$

While there are several variations for fold creation, the framework employs *stratified sampling* (Kohavi, 1995) to address the class imbalance. This technique involves dividing the data into folds based on the class of each example. A random sample is drawn from each fold, guaranteeing a proportional representation of all classes within each fold. This is critical when the

target variable’s values differ significantly between classes. By ensuring balanced folds, stratified sampling mitigates class imbalance’s impact on model performance and facilitates fair evaluation across all classes of the target variable.

The CreateFolds function (Equation 4.3) takes three arguments: the dataset (D), the number of folds (K), and the target variable (y). The target variable distribution is used for stratified sampling. The method creates a set containing K folds, each holding a subset of examples. Within the *Inmplode* framework, the folds of a dataset D_i have an initial index indicating the dataset to which they belong. Therefore, folds are denoted by $\{f_{i,1}, \dots, f_{i,K}\}$.

$$\{f_1, \dots, f_K\} \leftarrow \text{CreateFolds}(D, K, y) \quad (4.3)$$

The fold creation operation is an essential step in the framework’s workflow. As shown in Figure 4.2, it is performed only once for each of the N datasets, for consistency. These folds serve multiple purposes in subsequent framework tasks.

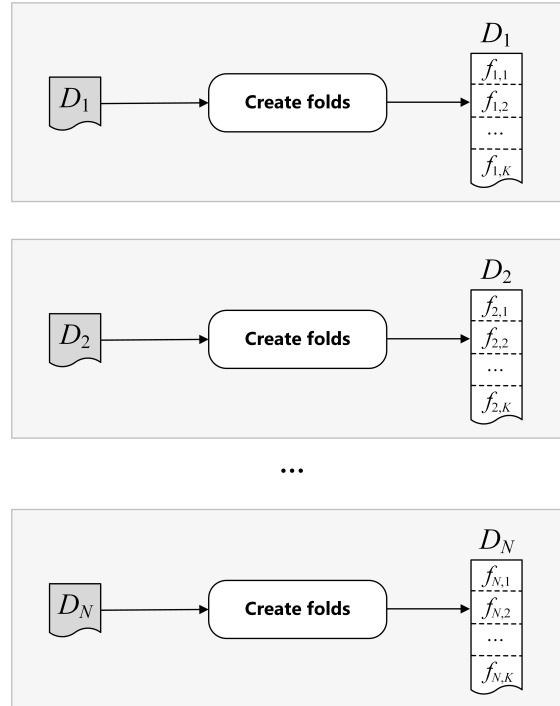


Figure 4.2: Creation of K folds for N datasets

Creation of folds in the case study

To balance bias and variance during model evaluation and considering the specific requirements of this case study, $k = 10$ was selected for the k -fold cross-validation. This choice is widely adopted in the machine learning community (Kohavi, 1995; Hastie et al., 2017). The following considerations further support the use of 10-fold cross-validation in

this specific context:

1. Minimum dataset size – The study only included courses with at least 100 enrolled students, ensuring a minimum of 100 examples per dataset. A higher minimum could have limited the number of courses analyzed, while a lower limit could have resulted in unreliable models and performance estimates. This restriction led to the creation of 730 models, encompassing 12% of the total 5779 courses.
2. Minimum positive class representation – Courses needed to have at least 10% of students failing (positive class). This minimum threshold ensures a baseline level of class imbalance. Stratified sampling guarantees that each fold, even in imbalanced datasets, contains at least one example from the positive class.

The case study used R’s ‘createFolds’ function for 10-fold cross-validation (R Core Team, 2023).

4.4 Training of operational models

Both operational model training and evaluation are performed within a setup, corresponding to an iteration based on k -fold cross-validation. Therefore, all tasks detailed next have a superscript specifying the setup (k) in which they are performed. There are two distinct contexts for training operational models in each setup:

- Deployment models (setup #0) – This setup is designed to create models for real-world use. Deployment models use the entire dataset as a training set ($Tr = D_i$), encompassing all available examples (Figure 4.3).

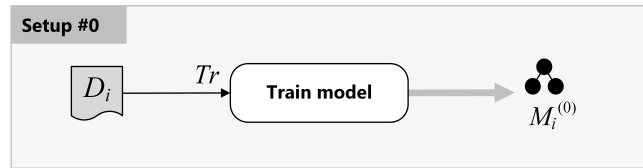


Figure 4.3: Training an operational model for deployment

- Evaluation models (setups #1 to #K) – These setups are designed to train models for evaluation. Unlike deployment models, each model uses the remaining folds other than the one associated with the setup k ($Tr = D_i \setminus f_{i,k}$) as training set (Figure 4.4).

The TrainModel function (Equation 4.4) creates a rule-based model. It requires as arguments the learning algorithm ($\mathcal{L}$), which specifies the model’s learning approach, encompassing details about the model format (e.g., decision tree, set of classification rules), and hyperparameter tuning options, the training set (Tr), the set of independent variables (X), and the target variable (y), resulting in a model M . Within the *Inmplode* framework, each operational model is denoted by $M_i^{(k)}$, indicating a model specific to dataset i within setup k .

$$M \leftarrow \text{TrainModel}(\mathcal{L}, Tr, X, y) \quad (4.4)$$

Training models in the case study

Due to their inherent interpretability, decision trees were selected as the rule-based models for this case study. Since the *Inmplode* framework relies on the decision table format for model merging, a conversion method is required. Section 5.2 details the process for converting decision trees to decision tables.

Decision trees were trained using the C5.0 method implementation from the R ‘caret’ package (Kuhn, 2008). This package automates hyperparameter tuning, ensuring the generated models are optimized for performance.

4.5 Evaluation of operational models

While traditional model evaluation often centers on predictive performance, the *Inmplode* framework adopts a more comprehensive approach. It encompasses a broader set of evaluation criteria, including interpretability, fidelity, and others, alongside predictive performance. The need for test data is contingent upon the specific criterion.

Operational model evaluation, similar to training, is performed within each experimental setup. The framework uses the `EvaluateOperationalModel` function (Equation 4.5) to evaluate a given model (M), based on specified criteria, ($\mathcal{Z}$), a designated test set (Ts), and the positive class ($\oplus$). The result is a set (Z) whose structure mirrors the evaluation criteria, enabling straightforward comparison and analysis. Within the framework, each set, $Z_i^{(k)}$, holds the evaluation results for the corresponding model version, $M_i^{(k)}$, with fold $f_{i,k}$ used as the test set.

$$Z \leftarrow \text{EvaluateOperationalModel}(M, \mathcal{Z}, Ts, \oplus) \quad (4.5)$$

Figure 4.4 depicts the evaluation of a model across all setups using the k -fold cross-validation method. Focusing on dataset D_i , the `TrainModel` function is used to create a model using all examples from D_i except for those belonging to the current fold, $f_{i,k}$. Subsequently, the trained model, along with the test data, is passed to the `EvaluateOperationalModel` function. Certain evaluation metrics, especially those requiring unseen data for testing, use the same excluded fold, $f_{i,k}$, as the test set. The setup-specific results are held in corresponding sets denoted by Z_i^k , where i refers to the dataset index and k refers to the fold number.

The final evaluation result is obtained by averaging the intermediate evaluations from each setup at the end of the process. This step occurs after developing and creating all models in the context of preparing the final output results. The results from all setups are aggregated in the results of setup #0 (hence the omission of this step in Figure 4.4).

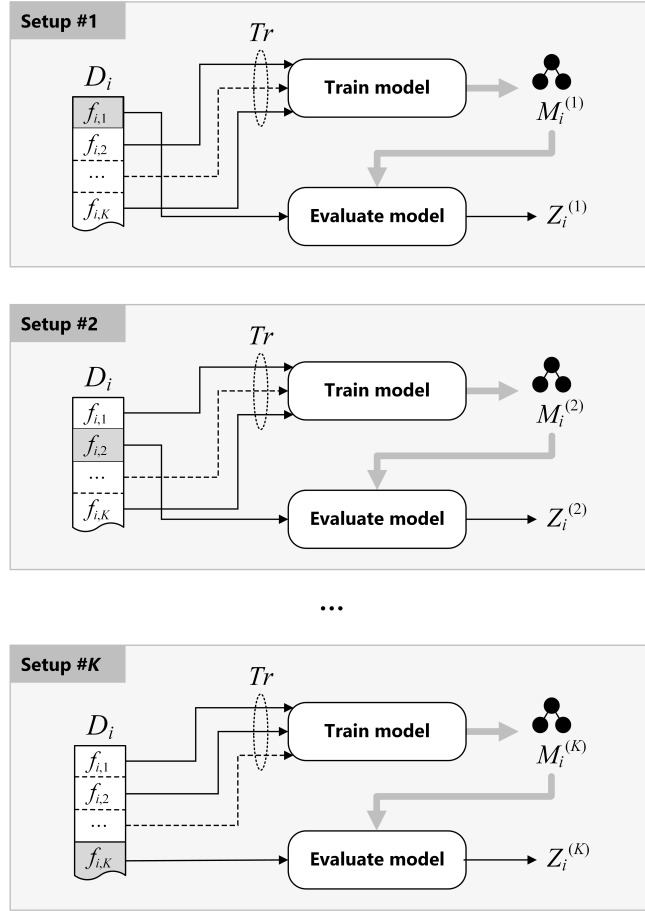


Figure 4.4: Training and evaluation of operational models

The set of model evaluation specifications ($\mathcal{Z}$), contains $|\mathcal{Z}|$ distinct specifications used to evaluate the performance of both operational and consensus models based on multiple criteria (Equation 4.6).

$$\mathcal{Z} = \{Z_1, \dots, Z_{|\mathcal{Z}|}\} \quad (4.6)$$

Each specification (Z_i) (Equation 4.7) is a tuple consisting of the evaluation criterion (e.g., performance, interpretability), the target models (operational, consensus, or both), and the set of evaluation metrics (Φ), each denoted as ϕ_i (Equation 4.8). An illustrative example of a specification is provided in Table 4.2.

$$Z_i = (\text{Criterion}_i, \text{Target}_i, \Phi_i) \quad (4.7)$$

$$\Phi = \{\phi_1, \dots, \phi_G\} \quad (4.8)$$

Algorithm 2 details the evaluation process for an operational model (M), given evaluation specifications, ($\mathcal{Z}$), a test set (Ts), and the positive class ($\oplus$). The algorithm iterates through each specification. For each specification applicable to operational models, an evaluation function (e.g., EvaluatePerformance or EvaluateInterpretability) is selected according to its specific criteria and designated metrics. The results related to the set of metrics (Φ_j) is collected into a set Z_j , which is

Table 4.2: Example of a model evaluation specifications set ($\mathcal{Z}$)

	Criterion	Target models	Φ
$\mathcal{Z}_1$	Performance	{Operational, Consensus}	{Accuracy, F1, MCC}
$\mathcal{Z}_2$	Fidelity	{Consensus}	{Accuracy, F1, MCC}
$\mathcal{Z}_3$	Interpretability	{Operational, Consensus}	{Size, Dimensionality, Depth}

then included in the overall evaluation results set for all criteria ($\mathcal{Z}$). The following sections detail the two evaluation criteria in Algorithm 2: performance and interpretability.

Algorithm 2 EvaluateOperationalModel: Evaluate an operational model from multiple criteria

Input:

- Model: M
- Evaluation specifications: $\mathcal{Z} = \{\mathcal{Z}_1, \dots, \mathcal{Z}_{|\mathcal{Z}|}\}$
- Test set: Ts
- Positive class: $\oplus$

Output:

- Results of evaluation: $Z = \{Z_1, \dots, Z_{|\mathcal{Z}|}\}$

for j such that $1 \leq j \leq |\mathcal{Z}|$ **do** $\triangleright$ Iterate over each evaluation specification

if ‘Operational’ $\in$ Target $_j$ **then**

$\triangleright$ Evaluate the predictive performance of the operational model

if Criterion $_j$ = ‘Performance’ **then**

$Z_j \leftarrow \text{EvaluatePerformance}(M, \Phi_j, Ts, \oplus)$

end if

$\triangleright$ Evaluate the interpretability of the operational model

if Criterion $_j$ = ‘Interpretability’ **then**

$Z_j \leftarrow \text{EvaluateInterpretability}(M, \Phi_j)$

end if

 ...

end if

end for

4.5.1 Evaluation of performance

Algorithm 3 details the EvaluatePerformance function, which evaluates the predictive performance of a given model according to a set of metrics. This function requires inputs such as the model itself (M), a set of performance metrics (Φ), the test set (Ts), and the positive class ($\oplus$). The algorithm uses the model to predict the value of the target variable for each example in the test set. It then computes a *confusion matrix* (Table 2.3), which summarizes the model agreements and disagreements between its predictions and the actual values of the target variable. Subsequently,

it iterates through each metric provided, using the corresponding metric with the counts of the confusion matrix as input. The result of each metric ϕ_i evaluation is held in z_i and subsequently aggregated into a set (Z). Examples of performance metrics are Accuracy (Equation 2.12), Precision (Equation 2.13), Recall (Equation 2.14), F1 (Equation 2.15), κ (Equation 2.16) and MCC (Equation 2.18).

Algorithm 3 EvaluatePerformance: Evaluate the predictive performance of a model

Input:

- Model: M
- Set of evaluation metrics: $\Phi = \{\phi_1, \dots, \phi_G\}$
- Test set: $Ts = \{e_1, \dots, e_n\}$
- Positive class: $\oplus$

Output:

- Set of results of evaluation: $Z = \{z_1, \dots, z_G\}$

▷ Make predictions on Ts

for i such that $1 \leq i \leq n$ **do** ▷ n is the size of the test set

$\hat{y}_i \leftarrow \text{Predict}(M, e_i)$ ▷ Prediction of model M for example e_i

end for

▷ Compute the confusion matrix

$TP \leftarrow |\{e_i : e_i \in Ts \wedge y_i = \oplus \wedge \hat{y}_i = \oplus\}|$

$TN \leftarrow |\{e_i : e_i \in Ts \wedge y_i \neq \oplus \wedge \hat{y}_i = \oplus\}|$

$FN \leftarrow |\{e_i : e_i \in Ts \wedge y_i = \oplus \wedge \hat{y}_i \neq \oplus\}|$

$FP \leftarrow |\{e_i : e_i \in Ts \wedge y_i \neq \oplus \wedge \hat{y}_i \neq \oplus\}|$

for g such that $1 \leq g \leq G$ **do** ▷ Iterate over each of the G metrics

$z_g \leftarrow \phi_g(TP, TN, FN, FP)$ ▷ Evaluate performance using the metric ϕ_g

end for

Evaluation of operational models and variables importance score distribution in the case study

The case study evaluated the performance of operational models with F1. Figure 4.5a shows the F1 distribution for models created in setup #1. The figure reveals significant variation in performance across the models. Only around a quarter of the models achieve a score above 0.60. This suggests potential for improvement in model performance for this particular setup. Figure 4.5b uses depicts the variations in the score across all setups, revealing that the difference in the variance of scores is not significant between setups. The maximum and minimum values, as well as the mode, exhibit similar trends across all setups.

The case study used the C5.0 algorithm implementation from the R language (Kuhn and Quinlan, 2023) to generate decision trees. This choice fostered the direct analysis of variable importance through model inspection. The C5.0 algorithm incorporates the concept of *Importance score of a variable* (Quinlan, 1993). This metric allows for assessing each variable's contribution to predicting the target variable's value.

To calculate the importance score, C5.0 counts the number of examples used to create rules for each variable. This count, denoted as $|Tr_x|$, is obtained by summing the examples associated with all rules that include a variable x (Equation 4.11). This calculation uses an *indicator function* (Gerber et al., 2003). This function assigns a value of 1 if a specific proposition (p) is true and 0 otherwise, translating logical comparisons into numerical values (Equation 4.9).

$$I[p] = \begin{cases} 1 & \text{if } p \text{ is true} \\ 0 & \text{otherwise} \end{cases} \quad (4.9)$$

The rule induction method, which generates decision trees or classification rules, assumes each example contributes to a single rule. Considering a model trained from a training set Tr with R rules, the sum of the number of examples (denoted as $|E_i|$) associated with each rule equals the total size of the training dataset (Equation 4.10).

$$|Tr| = \sum_{i=1}^R |E_i| \quad (4.10) \quad |Tr_x| = \sum_{i=1}^R (|E_i| [x \in X_i]) \quad (4.11)$$

The importance score of a variable, denoted as I_x^M for variable x in model M , reflects its contribution to the model's decision-making process (Equation 4.12). This score is calculated as the proportion of training examples used to create rules containing the variable. The variable used in the first split of the decision tree inherently has an importance score of 100%, as all examples contribute to its creation. Other variables might be used frequently in subsequent splits, but their importance score may be close to zero if the resulting terminal nodes only cover a small portion of the training set (Quinlan, 1993). To evaluate the use of independent variables across the models, the importance score of each variable (x) was averaged across all n operational models within a specific setup k (Equation 4.13).

$$I_x^M = 100 \frac{|Tr_x|}{|Tr|} \quad (4.12) \quad \bar{I}_x^k = \frac{1}{n} \sum_{i=1}^n I_x^{M_i^k} \quad (4.13)$$

The models used only a subset of the variables in the datasets. Figure 4.6a shows all variables' average importance score distribution for models created in setup #1. Figure 4.6b depicts the distribution of I_x across all setups. The most used variables are 'Delayed courses', 'Years of enrollment', 'Age', and 'Type of admission'. The variables 'Nationality', 'Status of student', 'Type of student', and 'Marital status' are much less used. The remaining variables are not used in any model.

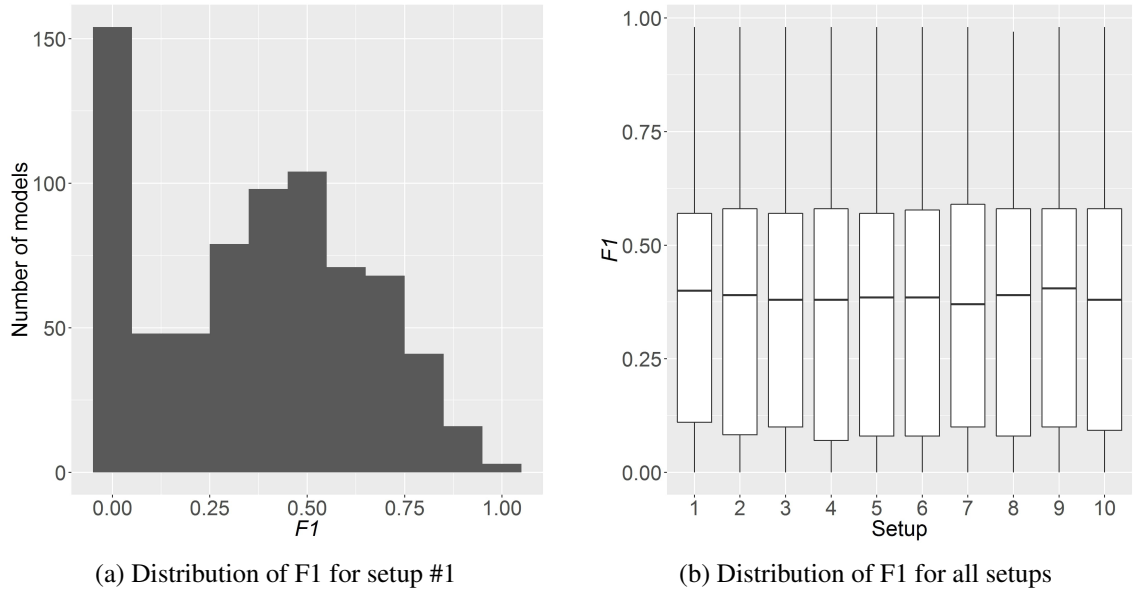


Figure 4.5: Model performance across different setups in the case study

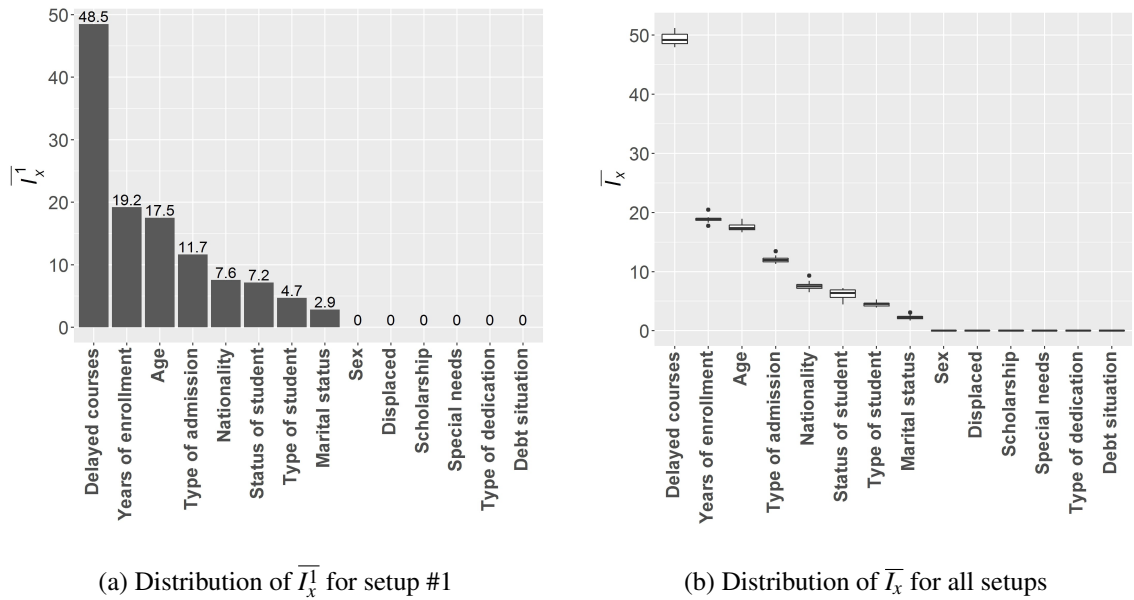


Figure 4.6: Variable importance distributions across different setups in the case study

4.5.2 Evaluation of interpretability

While predictive performance measures how accurately a model makes predictions on unseen data, interpretability refers to how easily a human can understand the reasoning behind the model's decisions. Unlike predictive performance, interpretability is an inherent property of the model itself, independent of any test set or prediction process. Algorithm 4 details the EvaluateInterpretability function, designed specifically for rule-based models. This function relies on a predefined set of metrics, denoted as (Φ) , to quantify different aspects of interpretability.

For each metric ϕ_i in Φ , the function computes a corresponding value z_i , representing a specific aspect of the model's interpretability. These individual scores are then collected into a set Z , which is the function's output. Examples of interpretability metrics of rule-based models are Size (Equation 3.16), Complexity (3.17), Average Rule Length (Equation 3.18), Depth (Equation 3.19), and Dimensionality (Equation 3.20).

Algorithm 4 EvaluateInterpretability: Evaluate the interpretability of a rule-based model

Input:

- Model: M
- Set of evaluation metrics: $\Phi = \{\phi_1, \dots, \phi_G\}$

Output:

- Set of results of evaluation: $Z = \{z_1, \dots, z_G\}$

for g such that $1 \leq g \leq G$ **do** $\triangleright$ Iterate over each of the G metrics
 $z_g \leftarrow \phi_g(M)$ $\triangleright$ Evaluate interpretability using the metric ϕ_g
end for

4.6 Stratification of operational models

Once the operational models are created, they are stratified into strata (each stratum corresponding to a set of models). The goal is to group related models so that they can be merged into a consensus model. To create these strata, a specific criterion is chosen, which can be based on:

- Domain knowledge – Uses data of the problem domain to stratify models. For example, a corporation might stratify performance-sales models by world region (e.g., ‘North America,’ ‘Europe,’ and ‘Asia’).
- Model description – Uses the models' properties as the basis for stratification. For example, stratifying models based on results ranges in one or more evaluation metrics.

Regardless of the criteria selected, the stratification process creates multiple sub-groups referred to as *strata*. Each group contains operational models with similar characteristics based on the chosen criteria. These characteristics are specified by a set of stratification variables denoted

by $\mathcal{V}$ (Equation 4.14). Each variable within this set is denoted by ε_j . The domain of each stratification variable represents the set of all possible values it can take (Equation 4.15). Each value within a domain is denoted by v .

$$\mathcal{V} = \{\varepsilon_1, \dots, \varepsilon_{|\mathcal{V}|}\} \quad (4.14) \quad \text{dom } \varepsilon_j = \{v_1, \dots, v_{|\varepsilon_j|}\} \quad (4.15)$$

The number of strata, denoted by J , is determined by combining all possible values from the stratification variable domains. This can be calculated as the product of each variable's cardinalities, i.e., the number of unique values (Equation 4.16).

$$J = |\text{dom } \varepsilon_1| \times \dots \times |\text{dom } \varepsilon_{|\mathcal{V}|}| \quad (4.16)$$

To illustrate the concept of stratification variables, consider a simple example with two stratification variables: $\mathcal{V} = \{\varepsilon_1, \varepsilon_2\}$, where ε_1 represents a world region, while ε_2 represents two intervals of model predictive performance measured by a metric (for example, F1).

- $\text{dom } \varepsilon_1 = \{\text{North America, Europe, Asia}\}$
- $\text{dom } \varepsilon_2 = \{[0.0; 0.7], (0.7; 1.0]\}$

This example demonstrates that the stratification variables can include numerical variables ($\mathcal{V}$). However, it is essential to note that numerical variables typically require an additional conversion step before they can be used for stratification. This conversion process transforms numerical values into discrete categories (intervals or bins). The score is converted into two categories based on a specific threshold in the example above.

Each variable represents a distinct dimension along which the models can be stratified. Combining the values of these variables results in a total of $J = 3 \times 2 = 6$ strata, as listed in Table 4.3.

Table 4.3: Example of strata created from stratification variable values

Stratum	ε_1	ε_2
S_1	North America	$[0.0; 0.7]$
S_2	North America	$(0.7; 1.0]$
S_3	Europe	$[0.0; 0.7]$
S_4	Europe	$(0.7; 1.0]$
S_5	Asia	$[0.0; 0.7]$
S_6	Asia	$(0.7; 1.0]$

A *stratum*, denoted as S (Equation 4.17), is a group of models that shares specific characteristics based on the chosen stratification criteria, defined by two elements:

- Stratification variable values set ($\mathcal{V}$) - Holds the values of a particular combination of the stratification variables ($\mathcal{V}$) that define the stratum. Each value in $\mathcal{V}$ corresponds to a particular combination category within the domain of a stratification variable.

- Model set ($\mathcal{M}$) – Represents the set of operational models that belong to the stratum. These models share the same characteristics defined by the $\mathcal{V}$ values. In other words, all models in $\mathcal{M}$ have stratification variable values that match the corresponding values in $\mathcal{V}$.

The subset of the operational models assigned to a stratum S_j is denoted as $\mathcal{M}_j$ (Equation 4.18). This subset contains L_j models from the total N models trained. The assignment of models to strata is mutually exclusive (a single model cannot belong to more than one stratum). To denote the stratum a set of models belongs to, an index j is included in the individual models.

$$S = (\mathcal{V}, \mathcal{M}) \quad (4.17) \quad \mathcal{M}_j = \{M_{j,1}, \dots, M_{j,L_j}\} \quad (4.18)$$

To perform the stratification process, the `StratifyModels` function (Equation 4.19) requires two inputs: a set comprising all operational models, denoted as $(M_1^{(k)}, \dots, M_N^{(k)})$, and a set of stratification variables ($\mathcal{V}$). The result is a set of strata $\mathcal{S}^{(k)}$, encompassing all the strata created for a setup k (Equation 4.20).

$$\mathcal{S}^{(k)} \leftarrow \text{StratifyModels} \left(\{M_1^{(k)}, \dots, M_N^{(k)}\}, \mathcal{V} \right) \quad (4.19)$$

$$\mathcal{S}^{(k)} = \{S_1^{(k)}, \dots, S_J^{(k)}\} \quad (4.20)$$

As depicted in Figure 4.7, within the context of a particular setup k , the process takes the N operational models and, using the specified stratification criteria, creates J strata of models.

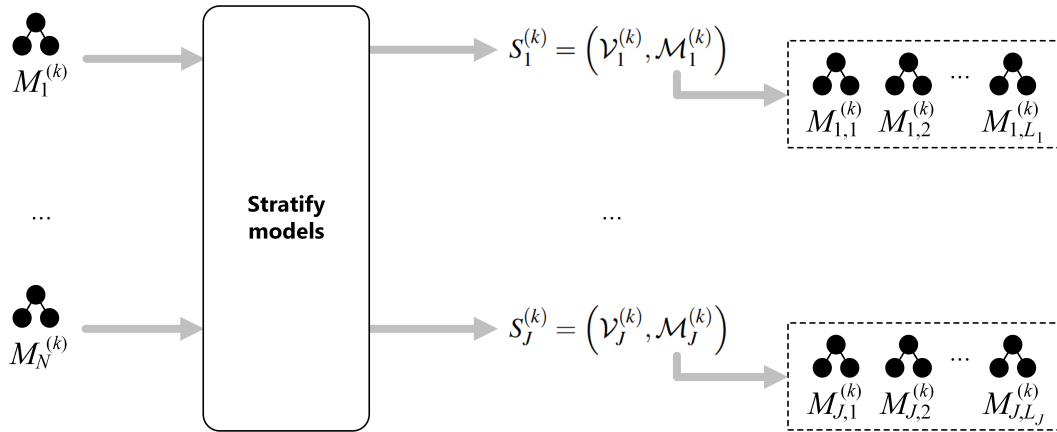


Figure 4.7: Stratification of operational models into multiple strata

Stratification of models in the case study

In the case study, stratification is based on using meta-variables, a set of characteristics used to describe the models themselves, separate from the standard variables within the models' training datasets. This distinction allows the creation of a model dataset (denoted as D_M). Each entry in this dataset represents a model alongside its corresponding meta-variable

values (corresponding to the stratification variables $\mathcal{V}$).

Table 4.4 exemplifies a model dataset from setup #1 in the case study. In this example, the models pertain to courses offered at a university. The domain knowledge meta-variables include the Scientific Area, Faculty, Program, and Course. The model description meta-variables include the predictive performance metric (F1) and two metrics related to interpretability (Size and Depth). While the domain knowledge meta-variables are the same for every evaluation model in k -fold cross validation, the model description meta-variables differ as each is trained in different training sets.

The case study employs three distinct criteria to create strata of the operational models:

- Domain knowledge – Groups models based on the scientific area of the courses they represent. It uses a real-world knowledge of the problem domain to categorize models into strata.
- Model complexity – Groups models based on the number of variables within each model. It reflects the model's overall complexity, recognizing that models with more variables may capture more intricate relationships within the data.
- Variable importance – Groups models based on the importance of variables, focusing on identifying the most influential variables for model performance.

Since the second and third criteria rely on the models' characteristics (model metadata), a clustering algorithm was employed to generate the strata. Specifically, the k -means algorithm (Hartigan and Wong, 1979) was used with k set to 4. This algorithm partitions the models into four strata based on similarities in their variable characteristics. Specifically, only the top four most important variables were considered for clustering, as identified by the average importance score in Figure 4.5b (e.g., Delayed Courses, Years of Enrollment, Age, Type of Admission). Each of these criteria results in a distinct set of strata. Figure 4.8 visually represents the outcomes of these three criteria for stratification alongside the ungrouped set of models for comparison.

The order in which operational models are combined can significantly influence the resulting consensus model, especially since the merging operation might not be commutative (meaning the order of combining models can affect the outcome). To explore this impact, the case study employs different criteria to order models within each stratum after stratification:

- Model complexity – Orders models based on the number of variables they contain. This aims to investigate whether combining simpler models first, followed by more complex ones, improves the final consensus model's quality. The number of variables in the models ranges from zero to seven. Models with the same number of variables are randomized in their combination order to avoid inherent bias.

Table 4.4: Models' dataset created in setup #1 in the case study

Model	Scientific area	Faculty	Program	Course	F1	Size	Depth
$M_1^{(1)}$	Mathematics	Sciences	Biology	Biostatistics	0.46	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$M_{730}^{(1)}$	Comp. Science	Engineering	Informatics	Logic	0.60	5	3

- Training data size – Orders models by the size of their training datasets. This differs from model complexity, as the number of examples does not directly determine complexity. The aim is to investigate if the order of merging models (those trained on smaller datasets first, then those on larger ones), impacts the consensus model's performance. Each course has a minimum enrollment of 100 students and a maximum of 950. As with the first criterion, models trained on the same data size are randomized in their combination order.
- Hierarchical clustering – Orders models using a hierarchical clustering method (Han et al., 2012). Like the third stratification criterion, the importance scores of the four most frequently used variables are used to calculate the Euclidean distance between models. While hierarchical clustering typically aims to create clusters (not used for this purpose), it also inherently arranges models based on their similarity (distance).
- Random – As a baseline for comparison, a final case involves not only no ordering criterion nor any kind of stratification. A single consensus model is generated.

4.7 Merging operational models into a consensus model

The model merging process begins by converting each model in a stratum into a corresponding decision table. The goal is to accommodate diverse model structures, such as decision trees or rule sets, allowing similar processing within the framework.

A merging loop then iterates through the models. In each iteration, the current model, represented as a decision table, is merged with the current merged table (initialized with the first converted model). A successful merge updates the merged table. Conversely, a failed merge leaves the merged table unchanged, signaling incompatibility between the models. This iterative process continues until all models have been processed. The final merged table, representing the consensus model, is then converted back to the original format of the input models.

As the core process of the *Inmplode* framework, a detailed discussion of the merging process is the central focus of Chapter 5, delving into the various components of the process and how it can be customized.

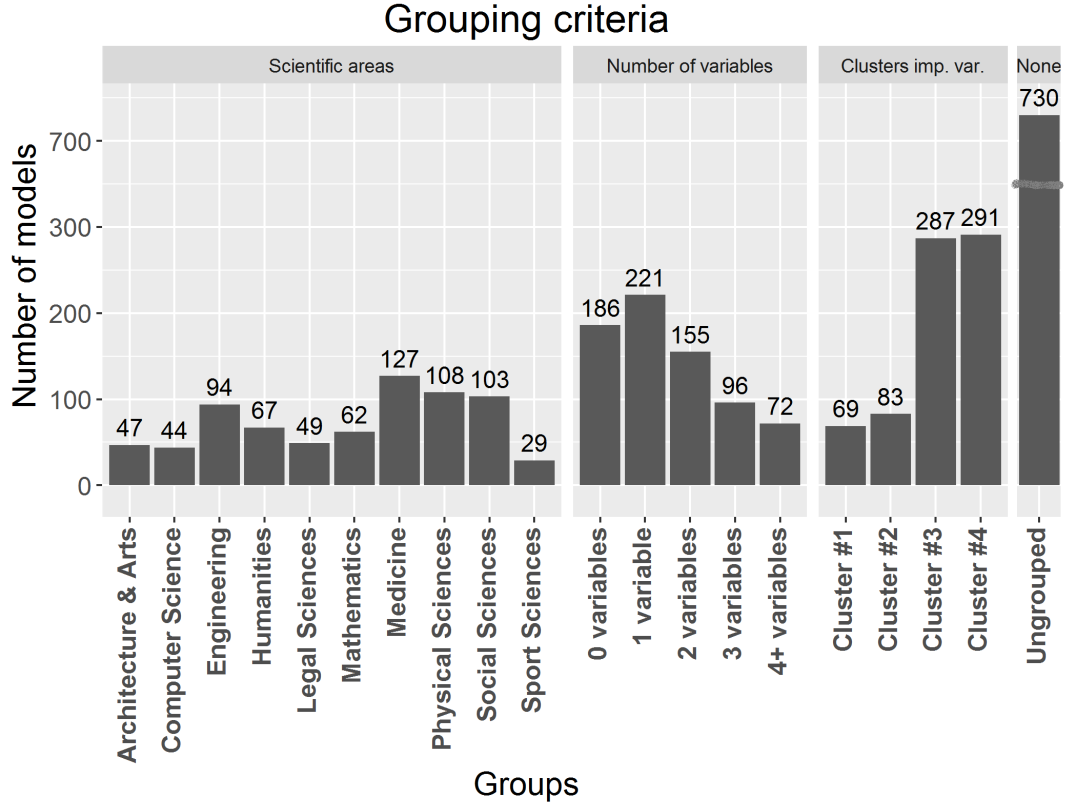


Figure 4.8: Models contained in groups across different grouping criteria (setup #1)

4.8 Evaluation of consensus models

Similar to operational models, consensus models require thorough evaluation from multiple criteria. While the evaluation metrics used for consensus models can be drawn from the existing set applied to operational models, fidelity is one key distinction, as it is specific to consensus models.

Algorithm 5 describes the evaluation process for a consensus model ($\bar{M}$), extending the approach used for operational models in Algorithm 2. Unlike the latter, it incorporates the set of operational models ($\mathcal{M}$) used to derive the consensus model, essential for fidelity evaluation, and a set of test sets ($\mathcal{T}$) rather than a single test set. Other inputs, such as evaluation metrics ($\mathcal{Z}$) and the positive class ($\oplus$), remain the same. The output is a set of metric values ($\bar{\mathcal{Z}}$), representing the results of the consensus model evaluation. To ensure alignment with the intended scope, the algorithm first checks if the metrics are applicable to consensus models. If not, it skips the corresponding evaluations; otherwise, it assigns them to specialized functions based on the evaluation criteria.

The evaluation encompasses four criteria: performance, performance loss, fidelity, and interpretability. Performance is assessed using the union of all subsets in $\mathcal{T}$ and follows the same algorithm as the operational models (Algorithm 3). Performance loss is evaluated within the same context as performance assessment, relying on its results along with the performance results of the operational models. Fidelity evaluation, however, retains the distinct subsets in $\mathcal{T}$ to preserve the

contextual information necessary for comparing the consensus model with individual operational models. Interpretability evaluation, which focuses on model characteristics such as the number of rules or variables used, does not require test sets and remains consistent for both operational and consensus models. It follows the same algorithm applied to operational models (Algorithm 4). The evaluation of performance, performance loss, and fidelity for consensus models is detailed in the following sections. Note that additional evaluation criteria can also be incorporated as needed.

Algorithm 5 EvaluateConsensusModel: Evaluate a consensus model from multiple criteria

Input:

- Consensus model: $\bar{M}$
- Set of operational models: $\mathcal{M} = \{M_1, \dots, M_L\}$
- Set of performance results of operational models: $\{Z_1, \dots, Z_L\}$
- Evaluation specifications: $\mathcal{Z} = \{\mathcal{Z}_1, \dots, \mathcal{Z}_{|\mathcal{Z}|}\}$
- Set of test sets: $\mathcal{T} = \{Ts_1, \dots, Ts_L\}$
- Positive class: $\oplus$

Output:

- Results of evaluation: $\bar{Z} = \{\bar{Z}_1, \dots, \bar{Z}_{|\mathcal{Z}|}\}$

```

for  $j$  such that  $1 \leq j \leq |\mathcal{Z}|$  do  $\triangleright$  Iterate over each of evaluation specifications
  if 'Consensus'  $\in$  Target $_j$  then
     $\triangleright$  Evaluate the performance and performance loss of the consensus model
    if Criterion $_j$  = 'Performance' then
       $Ts \leftarrow Ts_1 \cup \dots \cup Ts_L$ 
       $\bar{Z}_j \leftarrow \text{EvaluatePerformance}(\bar{M}, \Phi_j, Ts, \oplus)$ 
       $\bar{\Delta}_j \leftarrow \text{EvaluatePerformanceLoss}(\bar{Z}_j, \{Z_1, \dots, Z_L\})$ 
    end if

     $\triangleright$  Evaluate the fidelity of the consensus model with the operational models
    if Criterion $_j$  = 'Fidelity' then
       $\bar{Z}_j \leftarrow \text{EvaluateConsensusFidelity}(\bar{M}, \mathcal{M}, \Phi_j, \mathcal{T}, \oplus)$ 
    end if

     $\triangleright$  Evaluate the interpretability of the consensus model
    if Criterion $_j$  = 'Interpretability' then
       $\bar{Z}_j \leftarrow \text{EvaluateInterpretability}(\bar{M}, \Phi_j)$ 
    end if
  ...
end if
end for

```

4.8.1 Evaluation of performance

Traditional evaluation of consensus models often compared their predictive performance to a baseline model trained on all data using specific metrics (e.g., Accuracy). Studies exemplified this approach by demonstrating a slight baseline model improvement (around 1% improvement) (Hall et al., 1998), investigating how accuracy scales with the number of merging groups (finding similar performance even with sixteen groups) (Andrzejak et al., 2013), and comparing the consensus model’s test set classification accuracy with a decision tree directly induced from the entire dataset (revealing similar performance) (Bursteinas and Long, 2001).

Inmplode adopts the same evaluation approach for operational and consensus models. This evaluation is performed independently for each experimental setup, as detailed in Algorithm 3. Specifically, the metrics are configurable parameters, allowing for flexibility in performance evaluation. However, it is paramount to ensure metric alignment when the objective is to compare the predictive performance of operational models against consensus models in subsequent analyses.

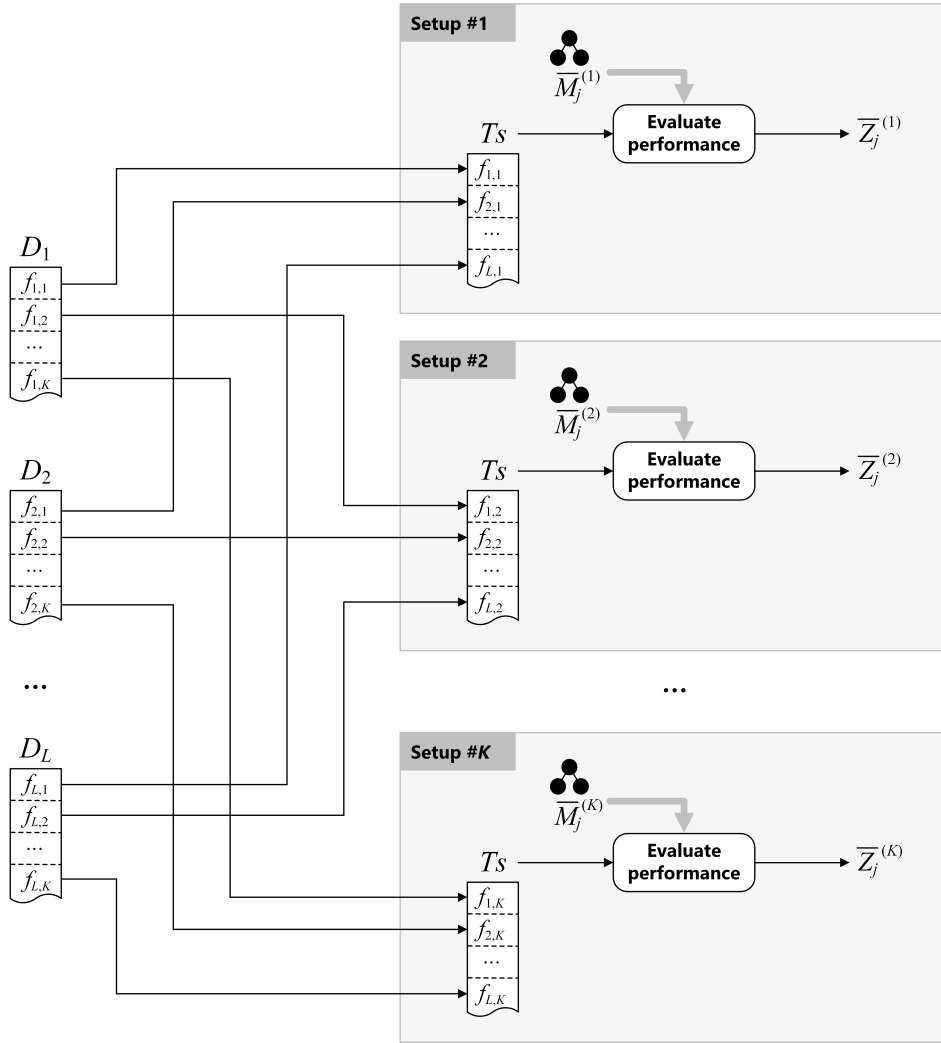
Figure 4.9 illustrates this evaluation setup. It highlights that the corresponding fold is reserved as test data for each setup to evaluate the consensus model for that setup. Since all L models were used to create each consensus model, all folds from each original dataset can be joined into a single test set for evaluation. Similarly to operational model performance evaluation, the result of each metric ϕ_i evaluation is held in $\bar{z}_i$ and subsequently aggregated into a set ($\bar{Z}$) that is returned by the function. For simplicity, the figure only shows the inputs related to consensus models and test sets. In the specific context of a stratum j containing L models, the test set for each setup k , denoted as T_s , is created from the union of all folds excluded from each dataset ($D_1, \dots, D_L$) used to train the operational models.

4.8.2 Evaluation of performance loss

Beyond evaluating the predictive accuracy of a consensus model in isolation, it is critical to quantify its performance in reference to the operational models it represents. For a given performance metric ϕ , the *Performance Loss*, denoted as $\Delta\phi$ (Equation 4.21), measures this difference. Specifically, $\Delta\phi$ compares the predictive performance of the consensus model ($\bar{M}$) against that of an operational model (M), using the latter as the baseline. This metric is generalizable to any performance measure and reflects how the consensus model deviates from the operational models.

$$\Delta\phi = \phi(\bar{M}) - \phi(M) = \bar{z} - z \quad (4.21)$$

The predictive performance of each pair of consensus model-operational model is evaluated in a shared test set. Performance loss is interpreted in reference to zero, a baseline indicating no change. A value of zero means that substituting the operational model with the consensus model yields equivalent predictive results. Negative values denote a performance decrease (the consensus model underperforms the operational model), while positive values reflect an improvement (the consensus model outperforms the operational model).


 Figure 4.9: Setups for evaluating the predictive performance of a consensus model for stratum j

In particular, the absolute difference ($\Delta\phi$) is prioritized over relative measures, such as percentage changes or ratios, to avoid division-by-zero errors. This scenario arises when the operational model performance metric, $\phi(M)$, is equal to zero, a scenario encountered in practical applications.

To formalize this evaluation, Algorithm 6 describes the process for calculating the performance loss of a consensus model using various performance metrics and operational models. The algorithm iterates over each metric and operational model, computes the performance loss values, and then averages these values across all L operational models to obtain the final measure, the *Average Performance Loss*, denoted as $\bar{\Delta\phi}$ (Equation 4.22). For consistency with other algorithms, the index g is used to iterate over the metric values. In the specific context of Algorithm 6, the Average Performance Loss is denoted by $\bar{\Delta}_g$.

$$\bar{\Delta\phi} \leftarrow \frac{1}{L} \sum_{i=1}^L (\phi(\bar{M}) - \phi(M_i)) \quad (4.22)$$

Algorithm 6 EvaluatePerformanceLoss: Evaluate the performance loss of a consensus model relative to a set of operational models

Input:

- Set of performance evaluation results for operational models: $Z = \{Z_1, \dots, Z_L\}$
- Performance evaluation results for the consensus model: $\bar{Z} = \{\bar{z}_1, \dots, \bar{z}_G\}$

Output:

- Performance loss results: $\bar{\Delta} = \{\bar{\Delta}_1, \dots, \bar{\Delta}_G\}$

▷ Compute the average performance loss across L operational models over G metrics

for g such that $1 \leq g \leq G$ **do**

$$\bar{\Delta}_g \leftarrow \frac{1}{L} \sum_{i=1}^L (\bar{z}_g - z_{i,g})$$

end for

4.8.3 Evaluation of fidelity

Fidelity evaluation shares some similarities with the approach used for evaluating the predictive performance of models. However, the distinction lies in the comparison target. While predictive performance evaluation compares a model's predictions against the actual values of a target variable, fidelity assessment focuses on comparing the predictions of two models. Specifically, it quantifies one model's ability to mimic another's predictions. This concept typically arises in contexts where a black-box model acts as the reference model, and a white-box model (usually a rule-based model) acts as the surrogate model (aiming to imitate the predictions and make them interpretable).

Within *Inmplode*, the reserved test fold serves to generate predictions from each dataset's operational model (serving as the reference) and its corresponding consensus model (acting as the surrogate). The comparison then centers on these paired predictions. Unlike standard performance evaluation, the operational model's prediction is considered the ground truth, allowing any evaluation metric to assess fidelity. Furthermore, incorporating a positive class enables the use of targeted metrics focused on how well the consensus model replicates the operational model's predictions, particularly for that class.

A key distinction between performance and fidelity evaluation lies in the use of test data. Performance evaluation uses a single, aggregated test set to evaluate the consensus model's predictive performance. In contrast, fidelity evaluation requires individual test sets for each operational model. Since each fold is trained on a specific data subset and uses a corresponding operational model for prediction, fidelity must be evaluated separately for each fold and its associated operational model. This approach provides a more granular evaluation of the consensus model's ability to replicate the behavior of individual operational models within a stratum. Although each fold and operational model is evaluated independently, a single consensus model is consistently applied across all evaluations within a stratum (Figure 4.10). This enables a direct comparison of the consensus model's fidelity to the predictions of each operational model within that stratum.

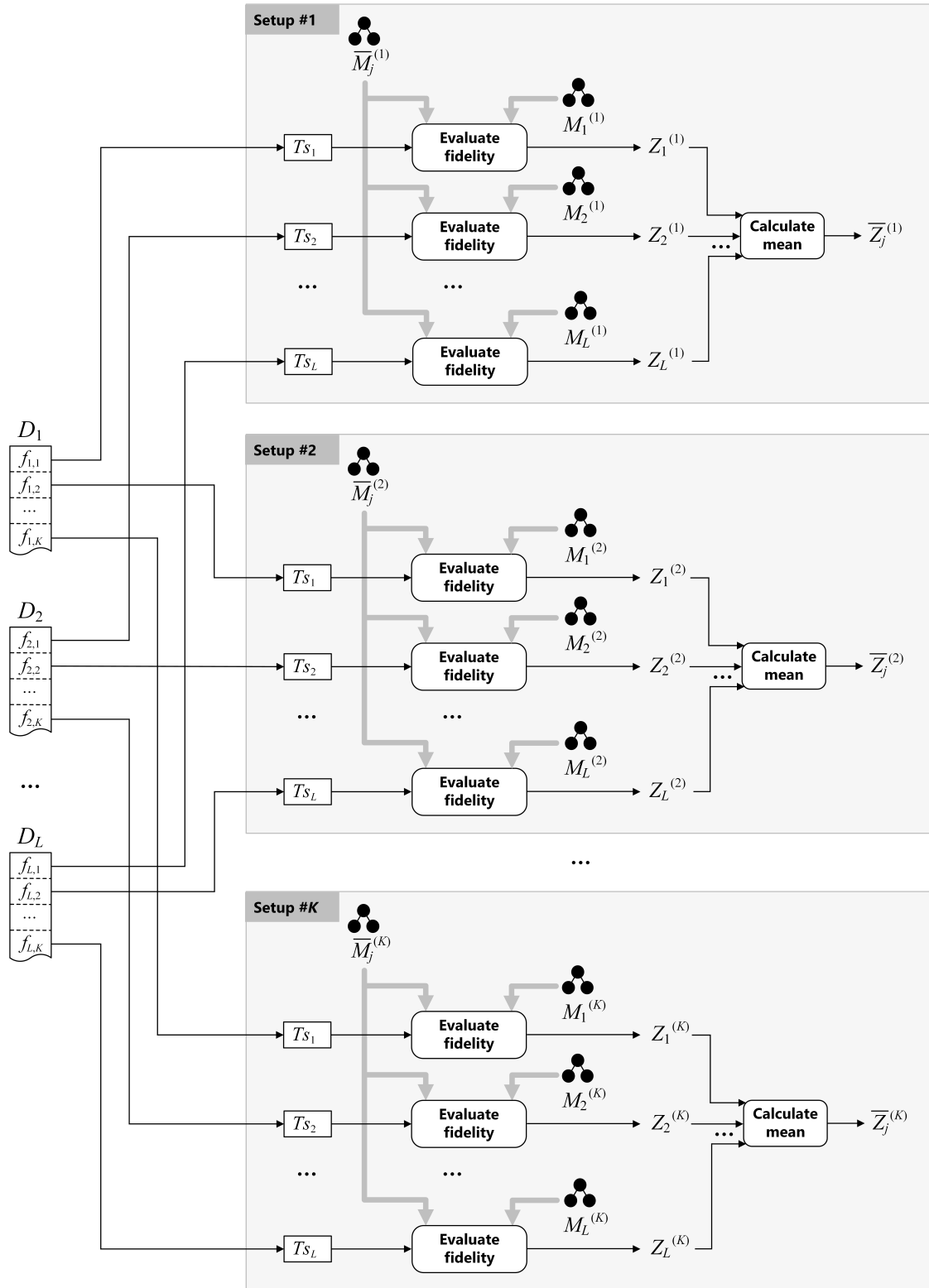


Figure 4.10: Setups for evaluating the fidelity of a consensus model for stratum j

Algorithm 7 details the process for determining the fidelity of a consensus model ($\bar{M}$) to a set of operational models ($\mathcal{M}$), using respective test sets ($\mathcal{T}$), evaluation metrics (Φ), and the positive class ($\oplus$). The EvaluateConsensusFidelity function calculates fidelity metrics for each operational model, using the consensus model as the surrogate and the operational model as the reference. To obtain a comprehensive measure of the consensus model's fidelity, these individual fidelity scores, each relative to an operational model, are averaged, with the results gathered in $\bar{Z}$.

Algorithm 7 EvaluateConsensusFidelity: Evaluate the fidelity of a consensus model relative to a set of operational models

Input:

- Consensus model: $\bar{M}$
- Set of models: $\mathcal{M} = \{M_1, \dots, M_L\}$
- Set of test sets: $\mathcal{T} = \{Ts_1, \dots, Ts_L\}$
- Set of evaluation metrics: $\Phi = \{\phi_1, \dots, \phi_G\}$
- Positive class: $\oplus$

Output:

- Set of results of evaluation: $\bar{Z} = \{\bar{z}_1, \dots, \bar{z}_G\}$

▷ Fidelity is evaluated with each model separately

for i such that $1 \leq i \leq L$ **do** ▷ L is the number of models

$Z_i \leftarrow \text{EvaluateFidelity}(\bar{M}, M_i, Ts_i, \Phi, \oplus)$

end for

▷ The consensus model's fidelity is the mean of the fidelity to each model

for g such that $1 \leq g \leq G$ **do** ▷ G is the number of metrics

$\bar{z}_g \leftarrow \frac{1}{L} \sum_{i=1}^L z_{i,g}$

end for

4.8.3.1 Evaluation of the fidelity of a surrogate model to a reference model

Algorithm 8 describes the process for evaluating the fidelity of a surrogate model relative to a reference model. While the process shares some similarities with evaluating the predictive performance of a single model, the distinction lies in comparing predictions from two models against a target variable instead of comparing a single model's prediction to the actual value of the target variable.

The algorithm takes as input a surrogate model (M_S), a reference model (M_R), a test set (Ts), a set of metrics to assess fidelity (Φ), and the positive class ($\oplus$). It outputs a set (Z) of performance results for each metric. As is typical in evaluation processes, the algorithm begins by predicting each example in the provided test set. For each example, two predictions are made: one using the reference model ($\hat{y}_i^R$) and one using the surrogate model ($\hat{y}_i^S$).

Next, to evaluate model agreement (as described in Section 3.1.4.2), the algorithm computes a confusion matrix using the reference model's predictions as the ground truth for each example.

Results are regarded as positive agreements (*PA*) when both models predict the positive class and negative agreements (*NA*) when both predict the negative class (corresponding to *TP* and *TN* in traditional evaluation). Disagreements occur when predictions differ: positive disagreements (*PD*) when M_R predicts positive and M_S negative, and negative disagreements (*ND*) in the opposite case (corresponding to *FP* and *FN*, respectively).

The remaining part of the process is similar to performance evaluation. The algorithm iterates through each metric in the provided set, using the corresponding metric method to calculate the metric's value from the confusion matrix. The result (z_g) of each metric evaluation (ϕ_g) is arranged in a set with all the metric evaluation results (Z).

Algorithm 8 EvaluateFidelity: Evaluate the fidelity of a surrogate model relative to a reference model

Input:

- Surrogate model: $M_S \triangleright$ Model evaluated for fidelity
- Reference model: $M_R \triangleright$ Model to provide predictions as reference for surrogate model
- Test set: $Ts = \{e_1, \dots, e_n\}$
- Set of evaluation metrics: $\Phi = \{\phi_1, \dots, \phi_G\}$
- Positive class: $\oplus$

Output:

- Set of results of evaluation: $Z = \{z_1, \dots, z_G\}$

$\triangleright$ Make predictions on Ts using each model

for i such that $1 \leq i \leq n$ **do** $\triangleright n$ is the size of the test set

$\hat{y}_i^R \leftarrow \text{Predict}(M_R, e_i) \triangleright$ Prediction of the reference model for example e_i

$\hat{y}_i^S \leftarrow \text{Predict}(M_S, e_i) \triangleright$ Prediction of the surrogate model for example e_i

end for

$\triangleright$ Compute the confusion matrix

$PA \leftarrow |\{e_i : e_i \in Ts \wedge \hat{y}_i^R = \hat{y}_i^S \wedge \hat{y}_i^R = \oplus\}|$

$NA \leftarrow |\{e_i : e_i \in Ts \wedge \hat{y}_i^R = \hat{y}_i^S \wedge \hat{y}_i^R \neq \oplus\}|$

$PD \leftarrow |\{e_i : e_i \in Ts \wedge \hat{y}_i^R \neq \hat{y}_i^S \wedge \hat{y}_i^R = \oplus\}|$

$ND \leftarrow |\{e_i : e_i \in Ts \wedge \hat{y}_i^R \neq \hat{y}_i^S \wedge \hat{y}_i^R \neq \oplus\}|$

for g such that $1 \leq g \leq G$ **do** $\triangleright$ Iterate over each of the G metrics

$z_g \leftarrow \phi_g(PA, NA, PD, ND) \triangleright$ Evaluate fidelity using the metric ϕ_g

end for

4.9 Collection of results

The final step of the framework involves consolidating the obtained operational and consensus models, along with their respective evaluation results. This step encompasses the following tasks:

- Operational models for deployment ($\mathcal{S}$) – Set that encompasses all operational models intended for deployment within the context of the J respective strata created during the process, generated from setup #0 (Equation 4.23).

$$\mathcal{S} = \mathcal{S}^{(0)} = \left\{ \mathcal{S}_1^{(0)}, \dots, \mathcal{S}_J^{(0)} \right\} \quad (4.23)$$

- Evaluation results of operational models ($Z_{\mathcal{S}}$) – Set of overall evaluation results of the operational models. It is obtained by the average of the evaluations of operational models across the K folds, organized by the J strata, performed by two tasks:
 - For each operational model i within a stratum, the average of its evaluation results across all setups is computed (Equation 4.24). This calculation uses the AverageEvaluations function, as outlined in Algorithm 9.

$$\bar{Z}_{j,i} \leftarrow \text{AverageEvaluations} \left(\left\{ Z_{j,i}^{(k)} : 1 \leq k \leq K \right\} \right) \quad (4.24)$$

- Assemble a set $Z_{\mathcal{S}}$ containing all the averaged evaluation results for the operational models in each stratum (Equation 4.25). Therefore, each element of this set represents the average evaluation for an operational model (index i) within a stratum (index j).

$$Z_{\mathcal{S}} = \left\{ \left\{ \bar{Z}_{j,i} : 1 \leq i \leq L_j \right\} : 1 \leq j \leq J \right\} \quad (4.25)$$

- Consensus models for deployment ($\overline{\mathcal{M}}$) – Set containing all J consensus models intended for deployment, generated from setup #0 (Equation 4.26).

$$\overline{\mathcal{M}} = \left\{ \overline{\mathcal{M}}_1^{(0)}, \dots, \overline{\mathcal{M}}_J^{(0)} \right\} \quad (4.26)$$

- Evaluation results of consensus models ($Z_{\overline{\mathcal{M}}}$) – Set of overall evaluation results of the consensus models. It is obtained by the average of the evaluations of consensus models across the K folds, performed by two tasks:
 - For each consensus model, the average of its evaluation results across all setups is computed (Equation 4.27). This calculation is performed using the AverageEvaluations.

$$\bar{\bar{Z}}_j \leftarrow \text{AverageEvaluations} \left(\left\{ \bar{Z}_j^{(k)} : 1 \leq k \leq K \right\} \right) \quad (4.27)$$

- Assemble a set $Z_{\overline{\mathcal{M}}}$ containing all the averaged evaluation results for the consensus models (Equation 4.28). Therefore, each element of this set represents the average evaluation for a consensus model (index j).

$$Z_{\overline{\mathcal{M}}} = \left\{ \bar{\bar{Z}}_j : 1 \leq j \leq J \right\} \quad (4.28)$$

Algorithm 9 AverageEvaluations: Averaging a model evaluation results across setups

Input:

- Results of evaluation for a model across setups: $\{Z^{(1)}, \dots, Z^{(K)}\}$

Output:

- Average of results of evaluation: $\bar{Z} = \{\bar{z}_1, \dots, \bar{z}_G\}$

for g such that $1 \leq g \leq G$ **do** $\triangleright G$ is the number of metrics

$$\bar{z}_g \leftarrow \frac{1}{K} \sum_{i=1}^K z_g^{(i)}$$

end for

4.10 Summary

This chapter presents a core contribution to the field of model merging: a framework, named *Inmplode*, that unifies diverse approaches through the identification of shared characteristics, the development of a standardized terminology, and the decoupling of method specification from its concrete implementation. The *Inmplode* framework offers a flexible and structured generic workflow capable of supporting a wide range of execution strategies.

This chapter provides a comprehensive description of the *Inmplode* framework, including its constituent generic workflow steps: fold creation, operational model training and evaluation, stratification, and consensus model generation for each stratum. While the general problem of developing consensus models from operational models is addressed, a detailed methodological exploration is deferred to Chapter 5 as a separate problem.

Chapter 5

Merging a Set of Rule-based Models into a Single Model

This chapter focuses on the core process of the *Inmplode* framework: the merging of multiple models into a single model. While framed within the context of merging operational models from a stratum into a consensus model, this process is applicable beyond this scope as a standalone problem. To highlight the broader applicability, the chapter employs the more general terms *base models* and *merged model*, thus avoiding *Inmplode*'s context-specific references to operational or consensus models. Similarly, the term *set of models* is used instead of stratum. Throughout the chapter, each section addresses a challenge in model merging as identified in the literature, formalizing the underlying concepts and algorithms. As in Chapter 4, the case study developed within the University of Porto is used throughout the chapter to illustrate how these concepts are put into practice, offering concrete examples of task specialization.

5.1 Merging a set of models into a single model

The process to merge a set of models into a single model is handled by the `MergeModels` function, detailed in Algorithm 10. This function taking as inputs a set with L models designated for merging ($\{M_1, \dots, M_L\}$), a set of rule evaluation metrics (Φ_r), and two sets of abstract functions.

The first set of abstract functions, denoted as $\mathcal{F}_C$, encompasses model conversion. These functions facilitate the integration of diverse rule-based models, regardless of their specific format (e.g., decision trees, classification rules, or other representations), by translating them into an internal representation based on decision tables. By using decision tables as a standardized format, the merging process is simplified. However, this approach requires two essential tasks to be performed by the conversion functions. First, *ConvertToTable* transforms each base model from its native format into an equivalent decision table. Second, *ConvertFromTable* reverses the operation, converting the merged model back into the original format of the operational models. This enables the merging process to work on a unique representation while preserving the interpretability of the base models.

The second set of abstract functions, $\mathcal{F}_M$, encompass merging methods for executing specific and adaptable tasks during the model merging process. This set includes: *MergePairRules*, which combines a pair of rules into a single rule; *EstimateRuleMeasures*, which calculates various evaluation metrics for a given rule; *FilterRule*, which removes rules deemed to be of low quality based on predefined criteria; *ResolveRuleConflict*, which addresses inconsistencies within individual rules; and *MinimizeTable*, which reduces the cardinality of the rule set through simplification and redundancy elimination. The intricacies of each method will be further detailed in subsequent sections.

The merging process begins by converting the first model (M_1) into a decision table, which is then assigned as the initial merged table, denoted as $\bar{T}$. Subsequently, a loop iterates through the remaining models (M_2 to M_L). In each iteration, the current model (M_j) is converted to a decision table (T_j) and merged with the current merged table ($\bar{T}$) with the MergeTables function. This yields a result (η) indicating either ‘Merged’ or ‘Unmerged’, and an intermediate merged table ($\bar{T}_I$). If successful ($\eta = \text{‘Merged’}$), $\bar{T}$ is updated with $\bar{T}_I$. In the case of all merging attempts failing, $\bar{T}$ remains unchanged from the initial table. Finally, the merged table ($\bar{T}$) is converted back to the base model’s format, yielding the merged model ($\bar{M}$).

Figure 5.1 illustrates an instantiation of the MergeTables method within a set of models. For simplicity, in the figure, it is assumed that all merging attempts are successful. The following sections will discuss the details of each process involved.

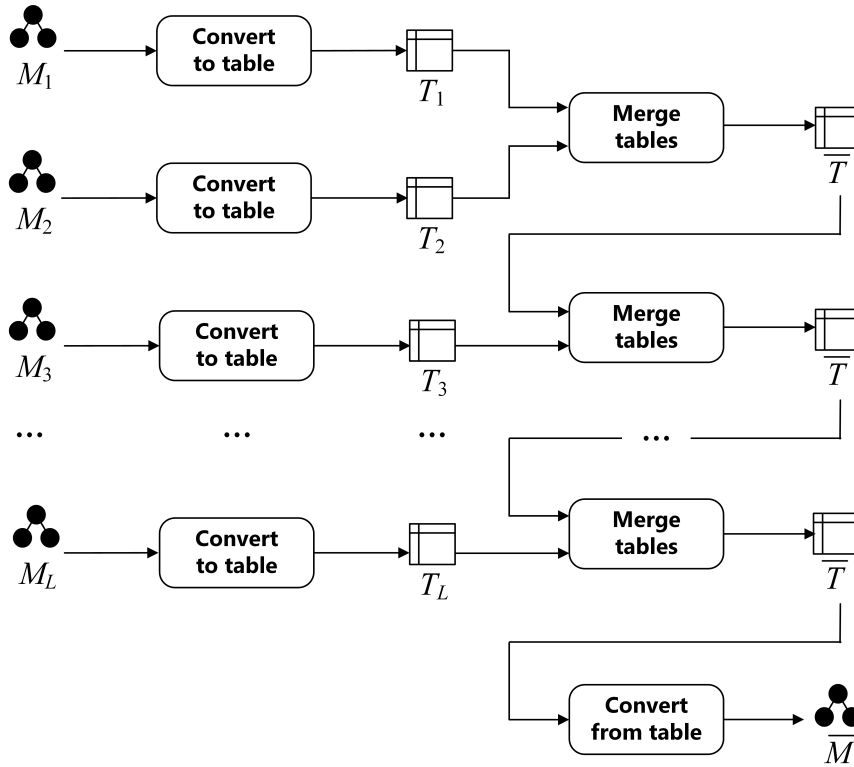


Figure 5.1: Merging a set of models into a single model

Algorithm 10 MergeModels: Merge a set of rule-based models

Input:

- Set of models: $\{M_1, \dots, M_L\}$
- Rule's evaluation metrics: Φ_r
- Conversion methods: $\mathcal{F}_C$
 - Convert a model to a decision table: *ConvertToTable*
 - Convert a decision table to a model: *ConvertFromTable*
- Merging methods: $\mathcal{F}_M$

Output:

- Merged model: $\overline{M}$

▷ Convert the first model to a decision table (initial current merged table)

$\overline{T} \leftarrow \text{ConvertToTable}(M_1, \Phi_r)$

for j such that $2 \leq j \leq L$ **do**

▷ Convert the subsequent model to a decision table

$T_j \leftarrow \text{ConvertToTable}(M_j, \Phi_r)$

▷ Attempt to merge the decision table with the current merged table

$\eta, \overline{T}_I \leftarrow \text{MergeTables}(T_j, \overline{T}, \mathcal{F}_M)$

if $\eta = \text{'Merged'}$ **then**

$\overline{T} \leftarrow \overline{T}_I$ ▷ The table of a successful merge attempt becomes the current merged table

end if

end for

▷ Convert the merged table to the format of the models in the set

$\overline{M} \leftarrow \text{ConvertFromTable}(\overline{T})$

5.2 Converting a model to a decision table

The merging process of models relies on their representation as decision tables. This approach offers a consistent and structured format for merging rule-based models regardless of their original format. The terminology surrounding decision table representations differs in the literature. Common terms include *rule set* (Hall et al., 1998), *hypercubes* (Bursteinas and Long, 2001), or *sets of iso-parallel boxes* (Andrzejak et al., 2013). An *Extended Attributive Decision Table* (Ligęza, 2006) is adopted for a formal definition. In this format, the model becomes a table where rows represent rules, columns represent variables, and cells contain sets of values (Table 2.5).

A straightforward approach to illustrate the model merging process involves converting a decision tree into a decision table. This conversion is considered lossless, meaning there is no loss of information during the process. In other words, the original decision tree and the resulting decision table represent the same underlying knowledge and logic for making decisions. This implies that

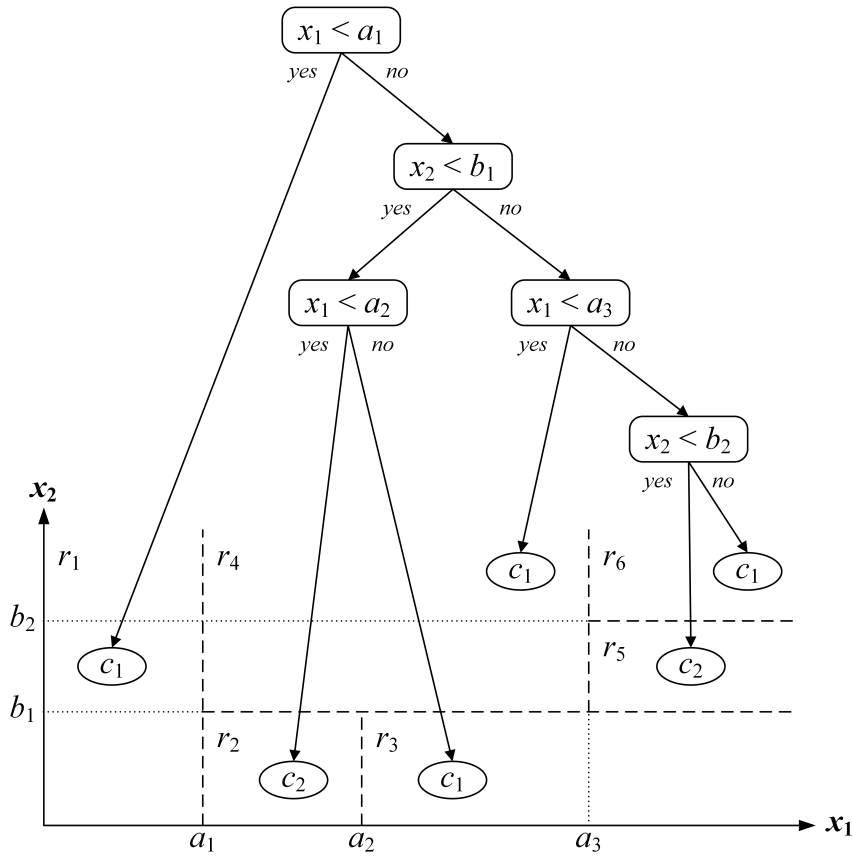


Figure 5.2: A decision tree model and underlying decision regions

any decision tree can be automatically converted into an equivalent decision table (Alonso Moral and Magdalena, 2011). In regular circumstances, the reverse conversion from a decision table back to a decision tree is also possible (Huysmans et al., 2011). In the conversion process, each path from the root node to a leaf node in the decision tree maps directly to a single row in the decision table. This means that each possible outcome has a corresponding row in the table. The conditions encountered along this path in the tree become the column headers in the table. The specific values that qualify for that path are also assigned to the corresponding cells in the table row. This ensures that all the information encoded in the decision tree, including conditions, branching logic, and outcomes, is preserved in the decision table.

Figure 5.2 provides an example of a decision tree in which each leaf node corresponds to a specific region within a two-dimensional decision space defined by the two independent variables, x_1 and x_2 . The boundaries of each region define a *subset*, which is projected onto the axis of each variable corresponding to the cells in the table that represent that specific rule and variable combination.

The ConvertTreeTable process starts at the root node of the decision tree. The initial condition evaluated at this node is $x_1 < a_1$. The outcome *yes* leads directly to a leaf node. Consequently, rule r_1 is defined within the subset $x_1 \in [0, a_1)$. The prediction of the target variable associated with this rule is c_1 , laid out in the first row of the decision table. In the outcome *no*, all subsequent regions

Table 5.1: Decision table derived from a decision tree

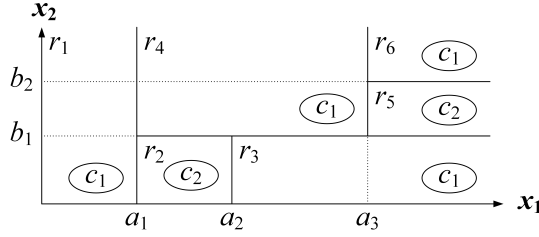


Figure 5.3: Decision regions derived from a decision tree

Rule	x_1	x_2	$\hat{y}$
r_1	$[0; a_1)$		c_1
r_2	$[a_1; a_2)$	$[0; b_1)$	c_2
r_3	$[a_2; +\infty)$	$[0; b_1)$	c_1
r_4	$[a_1; a_3)$	$[b_1; +\infty)$	c_1
r_5	$[a_3; +\infty)$	$[b_1; b_2)$	c_2
r_6	$[a_3; +\infty)$	$[b_2; +\infty)$	c_1

fall within the subset of the condition $x_1 \in [a_1, +\infty)$. The next condition evaluated is $x_2 < b_1$ at an intermediate node. For outcome *yes*, the rules are defined by the intersection of two subsets: $x_1 \in [a_1, +\infty)$ (inherited from the previous condition) and $x_2 \in [0, b_1)$. If the subsequent condition $x_1 < a_2$ has outcome *yes*, then rule r_2 is defined in the intersection of $x_1 \in [a_1, a_2)$ and $x_2 \in [0, b_1)$. This rule is represented in the second row of the decision table with a target variable value of c_2 . The process iterates through the remaining paths until all leaf nodes are reached. Figure 5.3 focuses on the decision regions defined by each rule, resulting in a row in the corresponding decision table (Table 5.1).

The conversion process also applies to models with a single decision rule. These models appear as decision trees with just one leaf node, missing any internal nodes (including the root), as shown in Figure 5.4. This single leaf node represents a rule covering the entire decision space. When converted to a decision table, this single-rule model translates to a table with a single row. Unlike typical decision tables, however, this row lacks columns for the independent variables (Table 5.5). Since the single rule applies universally, specifying conditions based on specific variables is redundant.

5.3 Expanded rule representation of a decision table

In its standard form, a rule is a conjunction of propositions determining a prediction (Equation 5.1). Each proposition comprises a check of whether the value of an independent variable falls within a

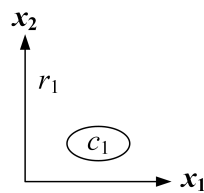


Figure 5.4: A single leaf decision tree

Figure 5.5: A single rule decision table

Rule	$\hat{y}$
r_1	c_1

specific set. Each set is a subset of the entire domain of its respective variable ($t_{i,j} \subset \text{dom } x_j$) and can be defined as a finite set of specific values, either categorical or numerical (Equation 5.2) or as an interval with a lower or upper bound for continuous numerical variables (Equation 5.3).

$$r_i : (x_1 \in t_{i,1}) \wedge (x_2 \in t_{i,2}) \wedge \dots \wedge (x_d \in t_{i,d}) \longrightarrow \hat{y}_i \quad (5.1)$$

$$t_{i,j} = \{c_1, c_2, \dots, c_j\} \quad (5.2) \quad t_{i,j} = [a; b] \quad (5.3)$$

To handle different tasks during the merging process, the representation of a rule described in Section 2.2 is extended with additional elements. Therefore, a decision rule (r_i) is formalized as a tuple (Equation 5.4), with elements as follows:

$$r_i = (X_i, h_i, \hat{y}_i, Z_i, \alpha_i, \beta_i, \sigma_i) \quad (5.4)$$

- Rule identifier (r_i) – A unique rule's identifier corresponding to its row position in the table (the subscript i indicates the rule's location in the i -th row).
- Independent variables (X_i) – The set encompassing the d independent variables that participate in the rule (Equation 5.5). To ensure consistency in the table representation, it is assumed that all rules within a given table share the same set of variables. If a specific variable does not have an associated subset of values, it is considered equivalent to the domain of that variable.

$$X_i = \{x_1, \dots, x_d\} \quad (5.5)$$

- Hyperrectangle (h_i) – The region of the decision space covered by the rule. The hyperrectangle of a rule, denoted as h_i , can be expressed as a tuple containing the subsets for each variable involved in the rule (Equation 5.6).

$$h_i = (t_{i,1}, \dots, t_{i,d}) \quad (5.6)$$

- Prediction ($\hat{y}_i$) – The class predicted by the rule for the target variable.
- Rule evaluation metrics results (Z_i) – A set with the evaluation results according to the rule evaluation metrics set (Φ_r) provided. Each metric within this set is denoted by ϕ_j (Equation 5.7).

$$\Phi_r = \{\phi_1, \dots, \phi_G\} \quad (5.7)$$

The evaluation result of metric j for rule r_i is a specific value $z_{i,j}$. These individual results are then compiled into Z_i (Equation 5.8).

$$Z_i = \{z_{i,1}, \dots, z_{i,G}\} \quad (5.8)$$

- Conflicting pair (α_i) – A pair of rules identified as having contradictory conclusions that require resolution before merging (Section 5.6).
- Joining set (β_i) – A set of rules identified as candidates for joining into a more general rule during the table minimization process (Section 5.7)
- Merge status (σ_i) – A descriptor indicating the outcome of an attempt to merge two rules (Section 5.4.2).

As discussed in Section 3.1.4.2, a decision table (T) can be represented by a tuple with two elements: the set of independent variables (X) corresponding to the table's columns and a rule set ($\mathcal{R}$), containing the rules, corresponding to the table's rows (Equation 5.9).

$$T = (X, \mathcal{R}) \quad (5.9)$$

When there is no ambiguity regarding the table to which a rule belongs, a rule and its elements are denoted by a single index (r_i), where i indicates the row position of the rule within that table. However, when it becomes necessary to analyze rules from different tables, the rules are assigned a two-part index (r_{ji}): the first component, j , identifies the table to which the rule belongs, and the second component, i , specifies the rule's row position within the decision table (Equation 5.10).

$$\mathcal{R}_j = \{r_{j1}, \dots, r_{jR}\} \quad (5.10)$$

Table 5.2 presents a decision table as a set of rules. It separates the independent variables into a common set (X) and the subset values for the rule hyperrectangles (h_i), along with the predictions ($\hat{y}$) and evaluation measures (z_i). However, when each rule is represented as a tuple (Equation 5.4), the variable set (X) is included within the rule (X_i) and is the same for all rules in the same table. Including X_i in each rule is essential when merging rules across different tables, as the X set can differ between decision tables. The remaining elements (α_i , β_i , and σ_i) are discussed within the context of the corresponding processes in subsequent sections.

Table 5.2: Expanded rule representation in a decision table

Rule	$X = \{x_1 \quad \dots \quad x_j \quad \dots \quad x_d\}$	$\hat{y}$	$\Phi_r = \{\phi_1 \quad \dots \quad \phi_G\}$
r_1	$h_1 = (t_{1,1} \quad \dots \quad t_{1,j} \quad \dots \quad t_{1,d})$	$\hat{y}_1$	$Z_1 = \{z_{1,1} \quad \dots \quad z_{1,G}\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_i	$h_i = (t_{i,1} \quad \dots \quad t_{i,j} \quad \dots \quad t_{i,d})$	$\hat{y}_i$	$Z_i = \{z_{i,1} \quad \dots \quad z_{i,G}\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_R	$h_R = (t_{R,1} \quad \dots \quad t_{R,j} \quad \dots \quad t_{R,d})$	$\hat{y}_R$	$Z_R = \{z_{R,1} \quad \dots \quad z_{R,G}\}$

Listing 5.1: A decision tree model trained with the C5.0 algorithm

```

delayed_courses <= 3: y (118/27)
delayed_courses > 3:
:...years_of_enrollment <= 2: n (15/1)
  years_of_enrollment > 2:
:...marital_status = married: n (4/1)
  marital_status = other:
:...delayed_courses <= 5: y (23/5)
    delayed_courses > 5:
      ...years_of_enrollment <= 4: n (12/2)
        years_of_enrollment > 4: y (13/4)

```

Converting from a decision tree to a decision table in the case study

This case study predicts whether a student will pass or fail a course. A decision tree (Listing 5.1) was trained using the C5.0 algorithm (Kuhn and Quinlan, 2023) on a course dataset. Besides the prediction itself, each leaf in the decision tree contains two numbers:

- Number of examples (B) – Indicates the number of examples used to create the rule;
- Number of misclassified examples ($\overline{HB}$) – Indicates the number of examples within the body of the rule incorrectly classified (when the prediction does not match the actual target variable value in the training set).

Table 5.3 displays the decision table derived from the decision tree. As specified in Table 4.1, x_3 corresponds to ‘Marital status’, x_{13} to ‘Years of enrollment’, and x_{14} to ‘Delayed courses’. To illustrate the calculation of metrics, the first rule presented in Table 5.3 is considered. The total number of examples, denoted by n , is obtained by summing the number of examples used in every rule. While a more extensive set of metrics could be considered (Section 3.1.4.1), this analysis focuses on the following metrics, which are directly extracted from the model: Number of examples ($\phi_1 = B$) and Number of misclassified examples ($\phi_2 = \overline{HB}$). Based on these, the following metrics can be calculated: Rule’s Coverage ($\phi_3 = \text{Coverage}$), as defined in Equation 3.6; Rule’s False Discovery Rate ($\phi_4 = \text{FDR}$), as defined in Equation 3.13; and Rule’s Weighted False Discovery rate ($\phi_5 = \text{WFDR}$), as defined in Equation 3.14. These metrics are computed during the model conversion process from a decision tree to a decision table. For each rule i , the resulting values are collected in Z_i . Adding the number of examples used in all rules results in $n = 185$. The evaluation metrics for the first rule are as follows:

- $z_1 = B_1 = 118$
- $z_2 = \overline{HB}_1 = 27$

Table 5.3: A decision table extracted from a C5.0 decision tree model

Rule	X			$\hat{y}$	Φ_r				
	x_{14}	x_{13}	x_3		B	$\overline{HB}$	Coverage	FDR	WFDR
r_1	$[0; 3]$			y	118	27	0.638	0.229	0.146
r_2	$(3; +\infty)$	$[0; 2]$		n	15	1	0.081	0.067	0.005
r_3	$(3; +\infty)$	$(2; +\infty)$	{married}	n	4	1	0.022	0.250	0.005
r_4	$(3; 5]$	$(2; +\infty)$	{other}	y	23	5	0.124	0.217	0.027
r_5	$(5; +\infty)$	$(2; 4]$	{other}	n	12	2	0.065	0.167	0.011
r_6	$(5; +\infty)$	$(4; +\infty)$	{other}	y	13	4	0.070	0.308	0.022

- $z_3 = \text{Coverage}_1 = \frac{B_1}{n} = 0.638$
- $z_4 = \text{FDR}_1 = \frac{\overline{HB}_1}{B_1} = 0.229$
- $z_5 = \text{WFDR}_1 = \frac{\overline{HB}_1}{n} = 0.146$

Table 5.3 highlights another issue: three subsets, $t_{1,13}$, $t_{1,3}$, and $t_{2,3}$, are not specified. This indicates that variables x_{13} and x_3 are unused in rule r_1 , and variable x_2 is unused in rule r_2 . As a result, these variables can take on any value without impacting the rule's prediction. This is equivalent to assigning the entire domain of the respective variable: $t_{1,13} = \text{dom } x_{13}$; $t_{1,3} = \text{dom } x_3$; and $t_{2,3} = \text{dom } x_3$. The merging process accounts for this property by treating unspecified subsets as encompassing the domains of the respective variable.

5.4 Merging two decision tables

Algorithm 11 outlines the process for merging two decision tables. This procedure is implemented by the MergeTables function, which accepts as input two decision tables, denoted as T_A and T_B , and a set of abstract functions, $\mathcal{F}_M$, to customize the merging operation. The function returns a result, represented by η_I , indicating whether the merge operation was successful, and a merged decision table, T (if the merge attempt is successful).

The merging process for two decision tables begins with the creation of a new rule-set ($\overline{R}$) by the MergeRuleSets method, which requires as inputs in addition to the rule sets of the two decision tables, the methods to merge rules (*MergePairRules*) and to estimate evaluation measures of a rule (*EstimateRuleMeasures*). The BuildMergedTable method evaluates the success of merging rules, ensuring that only rules meeting specified criteria (as determined by the *FilterRule* method) are included. If result in η_I is 'Merged' then the merge attempt was successful and the merged table is $\overline{T}$. The process proceeds to resolve any conflicts in the target variable using the ResolveRulesConflicts method (with the resolution strategy specified by *ResolveConflict*). Subsequently, an effort is made to simplify the table by reducing the number of rules with the *MinimizeTable* method. However, if the merging attempt results in an 'Unmerged' outcome, it is considered unsuccessful.

Algorithm 11 MergeTables: Attempt to merge two decision tables

Input:

- Decision tables to merge: $T_A = (X_A, \mathcal{R}_A), T_B = (X_B, \mathcal{R}_B)$
- Merging methods: $\mathcal{F}_M$
 - Merge a pair of rules: *MergePairRules*
 - Estimate evaluation measures of a rule: *EstimateRuleMeasures*
 - Filter out low-quality rules: *FilterRule*
 - Resolve a conflict in a rule: *ResolveRuleConflict*
 - Reduce the number of rules: *MinimizeTable*

Output:

- Result of the merge attempt: η
- Merged table: $\bar{T}$

▷ Attempt to merge the rules-sets from each table

$\bar{\mathcal{R}} \leftarrow \text{MergeRuleSets}(\mathcal{R}_A, \mathcal{R}_B, \text{MergePairRules}, \text{EstimateRuleMeasures})$

▷ Attempt to build a table from the merged rule set

$\eta, \bar{T} \leftarrow \text{BuildMergedTable}(\mathcal{R}_A, \mathcal{R}_B, \bar{\mathcal{R}}, \text{FilterRule})$

if $\eta = \text{'Merged'}$ **then**

$\bar{T} \leftarrow \text{ResolveRulesConflicts}(\bar{T}, \text{ResolveRuleConflict})$ ▷ Resolve any conflicts between rules

$\bar{T} \leftarrow \text{MinimizeTable}(\bar{T})$ ▷ Attempt to reduce the size of the merged table

end if

Depending on its result, the decision table may or may not effectively represent the merged table of the two original tables and be suitable for use in other contexts. The process is detailed in the next sections.

5.4.1 Merging rules sets

Algorithm 12 outlines the MergeRuleSets function, designed to create a new rule set by merging the rules from two rule sets ($\mathcal{R}_A$ and $\mathcal{R}_B$). Additionally, it requires two abstract functions to customize the process: *MergePairRules* and *EstimateRuleMeasures*. As the rules are created from all combinations of rule pairs from both rule sets, the process involves two nested loops: the outer loop iterates through rules in $\mathcal{R}_A$, while the inner loop iterates through rules in $\mathcal{R}_B$, creating a new rule for each pair combination. In each iteration, the CreateRule function creates a new rule. These are assigned a composite index (r_{Ai-Bj}), where i denotes the rule's index in $\mathcal{R}_A$, and j denotes its index in $\mathcal{R}_B$. The results is a merged rule set ($\bar{\mathcal{R}}$), encompassing all the newly created rules.

Algorithm 12 MergeRulesSets: Merge the rule sets of two decision tables

Input:

- Rule-set with rules to merge: $\mathcal{R}_A, \mathcal{R}_B$
- Function to merge a pair of rules: *MergePairRules*
- Function to estimate evaluation measures of a rule: *EstimateRuleMeasures*

Output:

- Merged rule-set: $\overline{\mathcal{R}}$

▷ Create a rule from merging each pair of rules from both rule sets

```

for  $i$  such that  $1 \leq i \leq R_A$  do
  for  $j$  such that  $1 \leq j \leq R_B$  do
     $r_{Ai-Bj} \leftarrow \text{CreateRule}(r_{Ai}, r_{Bj}, \text{MergePairRules}, \text{EstimateRuleMeasures})$ 
  end for
end for

```

▷ Merged rule-set with all pairs of rules from the original rule-sets

$\overline{\mathcal{R}} \leftarrow \{r_{Ai-Bj} : 1 \leq i \leq R_A, 1 \leq j \leq R_B\}$

5.4.2 Merging a pair of rules

Algorithm 13 details merging a pair of rules to create a new one. It takes as input the pair of rules being merged (r_a and r_b) and two functions: *MergePairRules* (the approach to merging a pair of rules) and *EstimateRuleMeasures* (to assign values corresponding to the evaluation metrics for the new rule).

Firstly, *CreateRule* defines the variables (X_m) and hyperrectangle (h_m) of the resulting rule with the *MergePairRules* method. Next, the function determines the prediction ($\hat{y}$). If both rules predict the same value, the prediction of r_a is assigned. Otherwise, a prediction conflict is signaled as ? to be resolved later. The *EstimateRuleMeasures* method sets the evaluation measures of the merged rule in Z_m . If there is a conflict, the conflicting rule pair is identified in α_m for later resolution. Finally, the process assigns a status (σ_m) based on the attempt's success. If all variable subsets in the merged rule are non-empty and there are no prediction conflicts, the attempt is considered 'Successful'. If all variable subsets are non-empty, but there's a disagreement in predicted target values between the original rules, it is signaled as a 'Conflict'. If at least one variable has an empty subset, regardless of any prediction conflicts, the attempt is regarded as 'Unsuccessful'.

5.4.3 Intersection of rules

An instantiation of the *MergePairRules* function is the intersection of two rules' hyperrectangles (Bursteinas and Long, 2001; Andrzejak et al., 2013; Strecht et al., 2014b). Algorithm 14 describes the process of intersecting two rules (r_a and r_b) resulting in a tuple containing the set of variables (X_m) and the hyperrectangle (h_m) of a new rule. The process begins by establishing that the rule variables (X_m) result from the union of the variables of the two rules to be merged. Next,

Algorithm 13 CreateRule: Create a rule by merging a pair of rules

Input:

- Rules to attempt to merge: r_a, r_b
- Function to merge a pair of rules: *MergePairRules*
- Function to estimate evaluation measures of a rule: *EstimateRuleMeasures*

Output:

- Rule created: $r_m = (X_m, h_m, \hat{y}_m, Z_m, \alpha_m, \sigma_m)$

▷ Set the variables and hyperrectangle of the rule

$$X_m, h_m \leftarrow \text{MergePairRules}(r_a, r_b)$$

▷ Set the prediction of the rule in case of rule agreement; otherwise, signal a conflict

$$\hat{y}_m \leftarrow \begin{cases} \hat{y}_a & \hat{y}_a = \hat{y}_b \\ ? & \hat{y}_a \neq \hat{y}_b \end{cases}$$

▷ Estimate the evaluation measures of the rule

$$Z_m \leftarrow \text{EstimateRuleMeasures}(r_a, r_b)$$

▷ Upon conflict, identify the conflicting rule pair for later resolution in the process

if $\hat{y}_m = ?$ **then**

$$\alpha_m \leftarrow (r_a, r_b)$$

end if

▷ Set the status of the merged rule

$$\sigma_m \leftarrow \begin{cases} \text{'Successful'} & \forall x_j \in X_m (t_j \neq \emptyset \wedge \hat{y} \neq ?) \\ \text{'Conflict'} & \forall x_j \in X_m (t_j \neq \emptyset \wedge \hat{y} = ?) \\ \text{'Unsuccessful'} & \exists x_j \in X_m (t_j = \emptyset) \end{cases}$$

the intersection of the hyperrectangles of each rule is performed in a loop that iterates through all the subsets of the variables of X_m . The resulting hyperrectangle will have all the variables from both rules. The subsets of each variable in the new rule ($t_{m,j}$) are calculated one variable at a time, leading to the following outcomes:

- Variables irrelevant in one rule – If a variable is not present in rule r_b or its subset in r_b equals its domain, the new rule incorporates the variable's subset from rule r_a . This means the variable is irrelevant for rule r_b . Conversely, if the variable is missing from rule r_a or its subset in r_a equals its entire domain, the new rule incorporates the variable's subset from rule r_b . Again, this indicates the variable is irrelevant for rule r_a .
- Variables present in both rules – If both variables are present in r_a and r_b , neither subset in their respective rules equals the entire domain (meaning the variable is relevant in both rules). The new rule's subset for that variable is the intersection of the subsets from rules r_a and r_b .

The intersection of subsets results in overlapping the two subsets and is a commutative operation ($t_a \cap t_b = t_b \cap t_a$). For subsets of numerical variables represented by intervals ($t_a = [a_1; a_2]$ and $t_b = [a_3; a_4]$), the resulting subset emerges from evaluating the maximum and minimum of the lower and upper bounds, respectively ($t_a \cap t_b = [\max(a_1, a_3); \min(a_2, a_4)]$). If the intervals do not overlap (e.g., $a_2 < a_3$), the resulting subset is an empty set ($t_a \cap t_b = \emptyset$), similar to when there are no common elements in two subsets of categorical variables.

Algorithm 14 IntersectRules (an instantiation of *MergePairRules*)

Input:

- Rules to merge: r_a, r_b

Output:

- Set of variables: X_m
- Hyperrectangle: h_m

▷ Set the variables of the merged rule

$X_m \leftarrow X_a \cup X_b$

▷ Intersect the subsets of both rules

$d \leftarrow |X_m|$

for j such that $1 \leq j \leq d$ **do**

$$t_{m,j} \leftarrow \begin{cases} t_{a,j} & x_j \in X_m \wedge (x_j \notin X_b \vee t_{b,j} = \text{dom } x_j) \\ t_{b,j} & x_j \in X_m \wedge (x_j \notin X_a \vee t_{a,j} = \text{dom } x_j) \\ t_{a,j} \cap t_{b,j} & x_j \in X_m \wedge x_j \in X_a \wedge x_j \in X_b \wedge t_{a,j} \neq \text{dom } x_j \wedge t_{b,j} \neq \text{dom } x_j \end{cases}$$

end for

▷ Set the hyperrectangle of the merged rule

$h_m \leftarrow (t_{m,1}, \dots, t_{m,d})$

For an intuition of the merging process by intersection, Figure 5.6 presents a geometrical depiction of the intersection of two models, model M_A (Figure 5.6a) and model M_B (Figure 5.6b). Therefore, both models have two numerical variables (x_1 and x_2) represented in a two-dimensional decision space. Each rule occupies a specific region of the decision space with an associated prediction (c_1 or c_2). Each model has a total of seven rules, r_{A1} to r_{A7} and r_{B1} to r_{B7} respectively.

Figure 5.6c shows the intersection result with the combinations of rules that share regions in the decision space, i.e., where the merge was successful, yielding model M_C . If the representation were in a decision table, the remaining combinations would have a status of unsuccessful (for example, the intersection of r_{A1} with r_{B6} results in the empty set).

An immediate consequence of intersection merging is a more fragmented decision space. The resulting model has 15 regions, even though each base model has only seven rules. This fragmentation increases the complexity of the model, making it harder to interpret. Furthermore, even with this relatively low number of regions, there are still 10 regions with prediction conflicts, denoted

by ?. These conflicts require resolution, as they represent areas where the two models disagree on the predicted outcome.

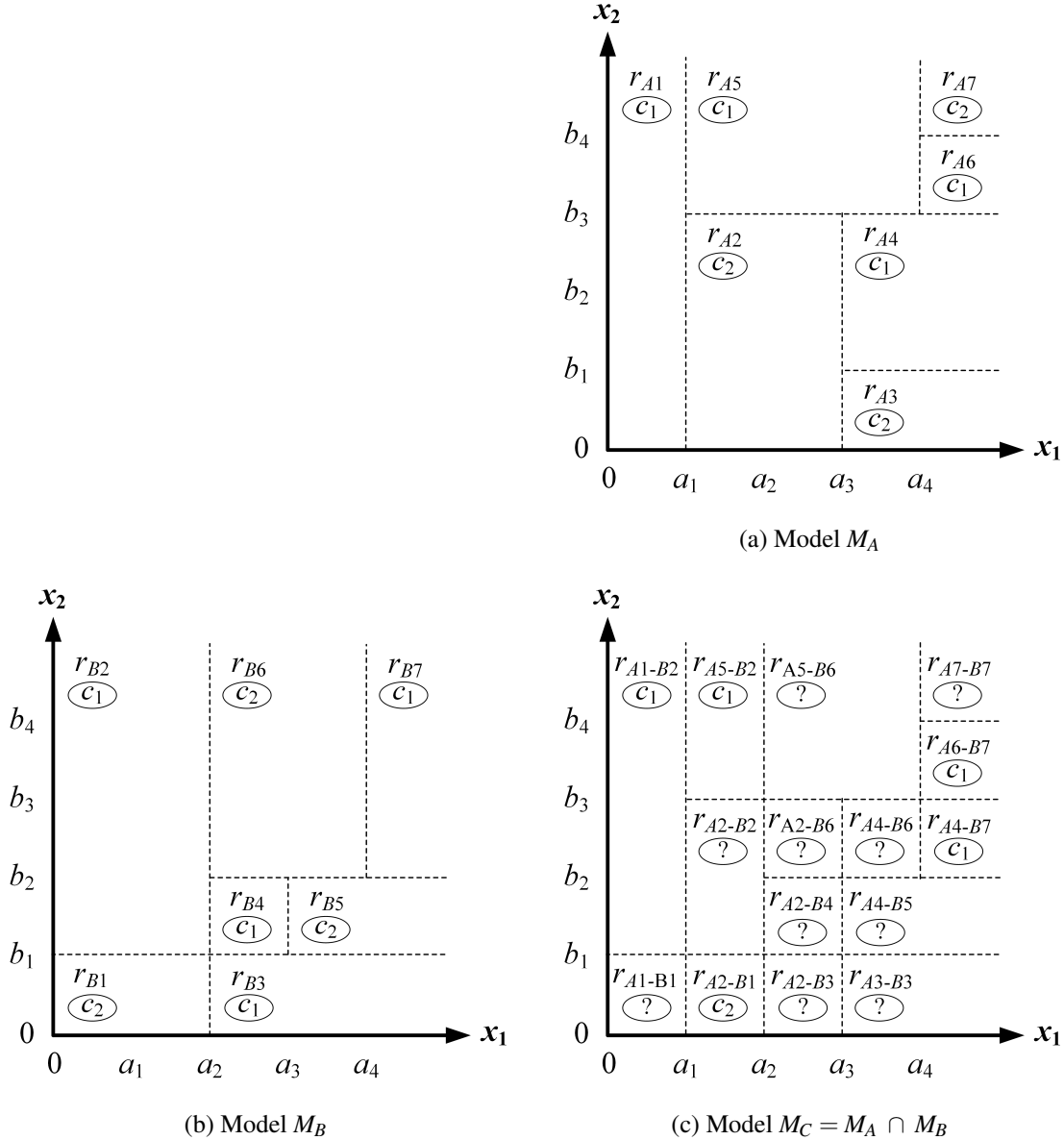


Figure 5.6: Geometrical depiction of merging two models by intersection

While this discussion has focused on rule intersection as a method for merging rules, it is important to acknowledge that other approaches can be employed. Fuzzy sets (Pedrycz and Gomide, 2007) offer a framework for alternative rule-merging techniques. Fuzzy sets are a generalization of classical sets (also known as crisp sets) where elements can have varying degrees of membership. This membership is represented by a value between 0 (not included) and 1 (fully included). Rules based on fuzzy sets form fuzzy rule-based models. However, for simplicity, it is assumed that the variables in the base models belong to crisp sets.

5.4.4 Estimation of the evaluation measures of a new rule

Estimating evaluation metrics for merged rules presents a distinct challenge, contrasting with the straightforward calculation possible in directly extracted decision tables. In the latter, metrics are readily derivable from individual rules owing to the availability of base values, such as coverage set size and the number of false positives. Merged tables, however, are devoid of this information, requiring the adoption of alternative methods for metric estimation. To address this challenge, the *EstimateRuleMeasures* function is employed to infer these values for the merged rules. This information is essential to select rules during the merging operation.

Algorithm 15 presents a simple approach to estimating the quality metrics of a merged rule. It calculates the average of the corresponding metrics from the two merged rules. This requires providing the two rules as input to the method. The method then calculates the average for each metric. Several heuristics can be employed to estimate metrics for merged rules. However, the essential issue is ensuring that the resulting values preserve the semantic meaning of the calculated metrics. In other words, the method designed to customize *EstimateRuleMeasures* should aim to produce metric values comparable to those obtained directly from rules learned by the base models. This ensures consistency when evaluating the quality of both learned and merged rules.

Algorithm 15 AverageMeasures (an instantiation of *EstimateRuleMeasures*)

Input:

- Rules to merge: r_a, r_b

Output:

- Resulting set with measures: $Z_m = \{z_{m_1}, \dots, z_{m,G}\}$

▷ Calculate the arithmetic mean between the measures in both rules for each metric

for j such that $1 \leq j \leq G$ **do** ▷ G is the number of metrics

$$z_{m,j} \leftarrow \frac{1}{2} (z_{a,j} + z_{b,j})$$

end for

Merging two models in the case study

Within the case study, the merging process is exemplified with decision tables T_A and T_B (depicted in Tables 5.4 and 5.5, respectively) along with their corresponding intersection table (Table 5.6). Therefore, the *MergeRulePair* method is customized by *IntersectRules* and *EstimateRuleMeasures* by *AverageMeasures*. Table T_A contains two rules that only use variable x_{14} ('Delayed courses'), while Table T_B contains three rules using two variables: x_1 ('Age') and x_{14} ('Delayed courses'). The merging process combines these tables to form a new decision table, $\bar{T}$, comprising six rules. Each rule in $\bar{T}$ represents merging one rule from T_A with one from T_B .

Table 5.4: Decision table T_A

Rule	x_{14}	$\hat{y}$	Coverage
r_{A1}	$[0; 1]$	y	0.68
r_{A2}	$(1; +\infty)$	n	0.32

Table 5.5: Decision table T_B

Rule	x_1	x_{14}	$\hat{y}$	Coverage
r_{B1}	$(36; +\infty)$		n	0.06
r_{B2}	$[0; 36]$	$[0; 2]$	y	0.83
r_{B3}	$[0; 36]$	$(2; +\infty)$	n	0.11

The hyperrectangle associated with each resulting intersected rule is calculated independently for each variable. This calculation involves finding the intersection of the hyperrectangles from the contributing rules for that specific variable. If the original subsets for a variable in the contributing rules have no overlap (disjoint sets), the resulting intersection is an empty set (denoted as $\emptyset$). This outcome is exemplified by the subset of variable x_{14} in rule r_{A1-B3} .

The target variable for an intersected rule is determined if its value remains consistent across both contributing rules, as exemplified in rules r_{A1-B2} , r_{A2-B1} , and r_{A2-B3} . However, a conflict arises when the contributing rules have different target values. This conflicting pair of rules can be represented as a tuple α . In such cases, the predicted value becomes unknown (denoted as $?$), as seen in rules r_{A1-B1} , r_{A1-B3} , and r_{A2-B2} . These conflicts are later addressed by the separate operation *Resolve conflicts*. The coverage of an intersected rule is estimated by averaging the coverages of the original rules.

This example illustrates how the absence of variables in one decision table can be addressed. When a variable is missing from one rule, its subset is transferred from the other rule into the resulting intersected rule. This is clear in rule r_{A1-B1} , where variable x_1 is absent in table T_A . Since there is no subset for this variable in table T_A , the subspace for x_1 is assumed from rule r_{B1} into the intersected rule. The rationale behind this approach is that missing variables act as neutral elements within the decision table. In other words, such a variable can take any value without influencing the prediction of the rule. This principle resembles how variables that have not yet been encountered are handled during the conversion of a decision tree to a decision table.

5.5 Building a merged decision table

Merging two decision tables does not always guarantee a successful outcome. To address this, the `BuildMergedTable` function (Algorithm 16) analyzes the created rules and determines the overall result of the merging process. The method takes as inputs the number of rules of the two original tables (R_A and R_B) that were merged, as well as the resulting merged rule-set ($\bar{\mathcal{R}}$), and an abstract method *FilterRule* to filter out low-quality rules. The method outputs are a result (η_I) indicating whether the merge was successful and a decision table ($\bar{T}$) that resulted from a successful merge.

The first step involves filtering the intermediate merged table to remove unsuccessful rule merge attempts. These rules, essentially empty sets, do not correspond to valid regions within the

Table 5.6: Merging decision tables T_A and T_B

Rule	x_{14}	x_1	$\hat{y}$	Coverage	α	σ
r_{A1-B1}	$[0; 1]$	$(36; +\infty)$	?	0.37	(r_{A1}, r_{B1})	Conflict
r_{A1-B2}	$[0; 1]$	$[0; 36]$	y	0.76		Successful
r_{A1-B3}	$\emptyset$	$[0; 36]$	?	0.40		Unsuccessful
r_{A2-B1}	$(1; +\infty)$	$(36; +\infty)$	n	0.19	(r_{A2}, r_{B2})	Successful
r_{A2-B2}	$(1; 2]$	$[0; 36]$	?	0.58		Conflict
r_{A2-B3}	$(2; +\infty)$	$[0; 36]$	n	0.22		Successful

decision space. Rules considered to be of low quality are also removed by the *FilterRules* (even if its status is ‘Conflict’, rendering the conflict irrelevant). After filtering, the table’s remaining k merged rules undergo renumbering. Consequently, rules identified by a combination of their original table identifiers (e.g., index r_{Ai-Bj}) are reindexed based on their row number (denoted as $\bar{r}_i$) within the merged table ($\bar{T}$).

The initial filtering process might result in a table with fewer rules or even no rules at all. If any rules remain, the next step involves normalizing the rule’s evaluation measures. This normalization is important to ensure that the quality metrics retain meaningful interpretation. In other words, this step guarantees that properties like the total coverage of the table remain one (while total coverage is an intrinsic property of models directly converted into tables, it might not hold after removing rules).

The next step is only performed if any rules remain in the decision table after filtering. It aims to identify and remove redundant variables due to rule elimination. The process analyzes the set of variables used by all the remaining rules. Rule removal might have eliminated rules that used specific variables, leaving those variables unused in the remaining set. This set of unused variables is denoted by X_{remove} . A variable is considered redundant if, for every remaining rule, the corresponding subsets always encompass the entire domain. All the redundant variables are removed from the variable sets of all remaining rules. The variables of the merged table are obtained from the union of the variables used in the rules.

The process concludes by defining the result of the intermediate merge attempt based on the existence or absence of rules after the removal process. The removal process may eliminate all the rules in the table, in which case the merge attempt is considered ‘Unmerged’. Conversely, if there are rules in the table, the result is ‘Merged’.

Research underscores the importance of retaining rules that encompass regions of high data density, effectively those containing a significant number of training examples (Andrzejak et al., 2013). Nonetheless, an overabundance of rules within the merged table can induce excessive fragmentation of the decision space, thereby diminishing interpretability. To mitigate this issue, specific instantiations of the *FilterRule* function can prioritize specific rule metrics. A few examples include Coverage, where only rules exceeding a predefined minimum coverage threshold are retained, as exemplified by the *RejectBelowCoverage* function (Algorithm 17); False Discovery

Algorithm 16 BuildMergedTable: Evaluate a merge attempt of two decision tables

Input:

- Number of rules on the original decision tables: R_A, R_B
- Attempted merged rule-set: $\overline{\mathcal{R}} = \{r_{A1-B1}, \dots, r_{A(R_A)-B(R_B)}\}$
- Method to filter out low-quality rules: *FilterRule*

Output:

- Result of the merge attempt: η
- Merged table: $\overline{T} = (X, \{\bar{r}_1, \dots, \bar{r}_k\})$

▷ Remove any unsuccessful or rejected rules and re-index the remaining ones

$k \leftarrow 0$

for i such that $1 \leq i \leq R_A$ **do**

for j such that $1 \leq j \leq R_B$ **do**

if $\sigma_{Ai-Bj} \neq \text{'Unsuccessful'} \wedge \text{FilterRule}(r_{Ai-Bj}) \neq \text{'Reject'}$ **then**

$k \leftarrow k + 1$

$\bar{r}_k \leftarrow r_{Ai-Bj}$

end if

end for

end for

▷ The intermediate merged decision table includes the rule set with the remaining rules

if $k > 0$ **then**

 ▷ Normalize the evaluation measures of the filtered rules

for j such that $1 \leq j \leq |\Phi_r|$ **do**

$Total \leftarrow \sum_{i=1}^k z_{i,j}$

for i such that $1 \leq i \leq k$ **do**

$z_{i,j} \leftarrow \frac{z_{i,j}}{Total}$

end for

end for

▷ Remove unused variables (where the subset equals the domain in all rules)

$X_{\text{remove}} \leftarrow \{x_j : \forall x_j \in X_i (t_{i,j} = \text{dom } x_j), 1 \leq i \leq k\}$

for i such that $1 \leq i \leq k$ **do**

$X_i \leftarrow X_i \setminus X_{\text{remove}}$

end for

▷ Set the variables of the merged table as the union of the variables used in the rules

$X \leftarrow \bigcup_{i=1}^k X_i$

end if

▷ Assign the process result as 'Merged' if the table contains rules, or 'Unmerged' otherwise

$\eta \leftarrow \begin{cases} \text{'Merged'} & k > 0 \\ \text{'Unmerged'} & k = 0 \end{cases}$

Table 5.7: Building a merged table

Rule	x_{14}	x_1	$\hat{y}$	Coverage	α	σ
r_1	$[0; 1]$	$(36, +\infty)$	?	0.18	(r_{A1}, r_{B1})	Conflict
r_2	$[0; 1]$	$[0, 36]$	y	0.36		Successful
r_3	$(1; +\infty)$	$(36, +\infty)$	n	0.09		Successful
r_4	$(1; 2]$	$[0, 36]$	?	0.27	(r_{A2}, r_{B2})	Conflict
r_5	$(2; +\infty)$	$[0, 36]$	n	0.10		Successful

Rate (FDR), where rules are selected based on a ratio of false positives to all predicted positives below a specified threshold; and Weighted False Discovery Rate (WFDR), which operates similarly to FDR but considers the entirety of the training set rather than exclusively the predicted positives, with a lower WFDR threshold indicating stricter filtering.

Algorithm 17 RejectBelowCoverage (an instantiation of *FilterRule*)

Input:

- Rule: r_m
- Coverage threshold: τ

Output:

- Decision: ‘Keep’ or ‘Reject’

▷ *Reject a rule below a specified coverage threshold otherwise, keep it*

$$\begin{cases} \text{‘Keep’} & \text{Coverage}_m \geq \tau \\ \text{‘Reject’} & \text{Coverage}_m < \tau \end{cases}$$

Building a merged table in the case study

Table 5.7 illustrates building a merged table in the case study. It exemplifies merging two tables by intersection (Table 5.6), where the coverage of the merged rules is set to the mean. Specifically, rule r_3 has a coverage of 0.09, which falls below the specified rejection threshold of 0.1 for rule removal. However, it is important to note that this value results from the normalization process applied to the metrics.

5.6 Handling rule’s conflicts

Decision table consistency, a critical factor in ensuring their quality (Ligeza, 2006), guarantees the absence of ambiguities and conflicts within a set of rules. Conflicts arise when multiple rules activate for the same example but predict different outcomes. Unlike decision trees, in which rules are mutually exclusive, decision tables may not inherently have this property. Therefore,

maintaining consistency in decision tables is essential for reliable decision-making. Well-defined rule priority schemes can be implemented to ensure that only one rule fires for a given example, thereby preventing indeterminism. It is important to note that another critical factor is completeness, which may be compromised if no rule activates for a given example. However, this issue will be addressed later in the process.

Conflicts arise in model merging when two rules from different tables cover the same hyperrectangle within the decision space (representing all possible combinations of conditions) but disagree on the predicted outcome. In the literature, there are several approaches to address this form of conflict, which include:

- Rule creation – This method eliminates conflict by creating new, more specific subrules from the conflicting ones. All newly created subrules would replace the originals (Alonso Moral et al., 2008). However, this can significantly increase the size and complexity of the decision table (Hall et al., 1998).
- Rule retraining and removal – This approach retrains a model using examples associated with the conflicting rules. If retraining eliminates the conflict, the predicted value from the retrained model is used. Otherwise, the conflicting rule is removed from the merged table. This strategy is limited to merging only two decision tables (Bursteinas and Long, 2001).
- Probability distribution – Instead of a single prediction at the leaf node, this approach uses a probability distribution for each possible outcome associated with each rule. Conflicts are resolved by assigning the class with the highest predicted probability (Andrzejak et al., 2013).
- Meta-rules – Meta-rules are rules where the prediction refers to another rule. While this can resolve conflicts, the rule-based system may require more context information about the decision and potentially introduce new variables for support (Al-Anbaki et al., 2019).
- Rule prioritization – This strategy prioritizes one rule over another based on a pre-defined criterion. The predicted value from the selected rule becomes the final prediction (Ligeza, 2006; Nalepa, 2018).

The `ResolveConflicts` function addresses conflicts identified during the merging process using rule prioritization (Algorithm 18). The function takes as inputs the original tables from which rules were merged (T_A and T_B), the resulting merged table ($\bar{T}$), and a `ResolveRuleConflict` abstract function that defines the specific conflict resolution criteria. This function is used for each merged rule identified as having a ‘Conflict’ status. For each conflicting pair, identified previously in α , it invokes the `ResolveRuleConflict` method, providing the pair as input. The first rule is obtained from $\alpha_{j,1}$, and the second rule from $\alpha_{j,2}$. This function determines the appropriate class to predict based on the defined criteria, resolving the conflict by returning the prediction of one of the rules. Once the conflict is resolved, the rule’s status in the merged table is changed to ‘Successful’.

Algorithm 18 Resolve conflicts

Input:

- Original decision tables: T_A, T_B
- Merged decision table: $\bar{T}$
- Method to resolve a conflict in a rule: *ResolveRuleConflict*

Output:

- Merged decision table without conflicts: $\bar{T}$

for j such that $1 \leq j \leq R_{\bar{T}}$ **do**

if $\sigma_j = \text{'Conflict'}$ **then**

$\hat{y}_j \leftarrow \text{ResolveRuleConflict}(\alpha_{j1}, \alpha_{j2}) \triangleright \text{Conflicting pair identified when merging the pair of rules}$

$\sigma_j \leftarrow \text{'Successful'}$

end if

end for

Algorithm 19 presents an instantiation of the *ResolveConflict* method based on selecting the rule with the highest coverage. In other words, this method prioritizes the rule that covers the highest percentage of examples in the training set based on its target variable. Table 5.8 demonstrates how this method resolves conflicts within the rules presented in Table 5.7. While this instantiation focuses on coverage, other approaches could involve a rule's *success factor* as selecting the rule with the lowest false discovery rate or weighted lowest false discovery rate. The conflict resolution strategy can use any of the results of one or more evaluation metrics from the original rules (captured during conversion to the decision table) to resolve conflicts.

Algorithm 19 RuleHighestCoverage (an instantiation of *ResolveRuleConflict*)

Input:

- Conflicting rules pair: r_a, r_b

Output:

- Prediction

$\triangleright$ Set the prediction of the rule with the highest coverage

$$\begin{cases} \hat{y}_a & \text{Coverage}_a \geq \text{Coverage}_b \\ \hat{y}_b & \text{Coverage}_a < \text{Coverage}_b \end{cases}$$

Rule's conflict resolution in the case study

Within the case study context, in Table 5.7, two decision rules, r_1 and r_4 , exhibit conflicts. Using the RuleHighestCoverage to resolve conflict, the coverages of the conflicting rules are compared. In the first case, this involves comparing $\text{Coverage}_{A1} = 0.68$ and

Table 5.8: Conflict resolution in merged table

Rule	x_{14}	x_1	$\hat{y}$	Coverage
r_1	[0; 1]	(36; $+\infty$)	y	0.18
r_2	[0; 1]	[0; 36]	y	0.36
r_3	(1; $+\infty$)	(36; $+\infty$)	n	0.09
r_4	(1; 2]	[0; 36]	y	0.27
r_5	(2; $+\infty$)	[0; 36]	n	0.10

$Coverage_{B1} = 0.06$ (Tables 5.4 and 5.5). Since $Coverage_{A1} > Coverage_{B1}$, the predicted target variable for r_1 becomes $\hat{y}_1 = \hat{y}_{A1}$. Similarly, for the second conflict, $Coverage_{A2} = 0.32$ and $Coverage_{B2} = 0.83$ are compared. As $Coverage_{A2} < Coverage_{B2}$, $\hat{y}_4 = \hat{y}_{B2}$. After resolving these conflicts, the status column (σ) can be omitted from Table 5.8 as all rules have successfully merged.

5.7 Minimizing the number of rules

While resolving rule conflicts ensures consistency in a decision table, other factors contribute to its overall quality (Ligeza, 2006) by minimizing redundancy, i.e., ensuring the table is free of unnecessary duplicate information. This can be achieved by implementing some or all of the following measures:

- Identifying and joining identical rules;
- Replacing less specific rules with more general ones (subsumption);
- Replacing multiple rules with a single equivalent rule (rule reduction);
- Eliminating unnecessary variables that become irrelevant after rule reduction.

Minimizing the number of rules within a decision table, while beneficial for both computational efficiency and human interpretability, presents a non-trivial challenge. Acknowledging the complexity of this task and the potential for diverse approaches, the problem of rule set minimization is abstracted into a *MinimizeRuleSet* function.

RuleGluing, presented in Algorithm 20 is an instantiation of the *MinimizeRuleSet* method. The algorithm takes the decision table to be minimized (T_C) as input and returns a potentially simplified table (T_D), i.e., with fewer rules as output. During this process, the algorithm iterates through the decision table, searching for pairs of rules with equal subsets in all variables except for one (the joined variable). When such a pair is found, the subsets of the joined variable can be joined using a union operation, effectively joining the rules into one. This process continues until no further minimization is possible. Although this approach bears a resemblance to the notion of joining

adjacent boxes (Andrzejak et al., 2013), other approaches can be found in the literature (Ligeza, 2006; Vermesan and Coenen, 1999).

More formally, the process aims to reduce the number of rules in the decision table for each iteration. It begins by setting the number of rules in the table at the start of the iteration to l . At the end of the iteration, this number is compared with the number of rules after the minimization attempt. Next, a joining set (denoted as β_i) is identified for each rule r_i . A *joining set* is a set of rules that includes r_i and all subsequent rules r_k , where $k > i$, that satisfy the following conditions:

- Shared prediction – All rules within a set must predict the same outcome (target variable value). The reference value is that of rule r_{Ci} , which is $\hat{y}_{Ci}$.
- Nearly identical hyperrectangles – Each rule in the set must have identical conditions (subsets) for all variables except one, which is referred to as the *joined variable* (x_j). This implies that the hyperrectangles of all rules are contiguous along one variable.

After this step, each rule has an associated joining set. Each rule in a joining set is denoted as $\beta_{i,j}$, where i denotes the rule to which the joining set is associated and j denotes the rule contained in the joining set (Equation 5.11).

$$\beta_i = \{\beta_{i,1}, \dots, \beta_{i,j}, \dots\} \quad (5.11)$$

Within the process, each joining set generates a rule in a new table, denoted as T_D . Therefore, each rule r_i from the original table T_C is scanned individually. The variable j controls the indices of the rules in the new table. A new rule (r_{Dj}) is included in the new table if it is either the first rule from the original table (T_C) or one of the subsequent rules not previously included in a joining set. If a new rule is selected, its construction begins by setting the set of variables to be the same as the first rule in β_i . Subsequently, the subsets of the rules belonging to the joining set are combined to create the hyperrectangle of the new rule (h_{Dj}). Essentially, the hyperrectangles within the set are joined based on their shared subsets, akin to being glued together.

The prediction for the new rule ($\hat{y}_{Dj}$) is the prediction of the first rule in the joining set since all rules in the joining set must have the same prediction. The measurements set associated with the evaluation metrics of the new rule (Z_{Dj}) are the direct sum of these measurements in the rules that make up the joining set. The new rule (r_{Dj}) has basic elements (set of variables, hyperrectangle, prediction, and evaluation measures). The remaining elements are not necessary.

After all the rules have been scanned, the minimized table is formed and copied to the reference table for a possible next iteration. Each iteration involves searching for new joining sets and creating corresponding merged rules. This process continues until no further number of rules is reduced between iterations. Reaching this point, the cycle ends with the simplified table T_D as output.

Algorithm 20 RuleGluing (an instantiation of *MinimizeRuleSet*)

Input:

- Decision table to minimize: T_C

Output:

- A (potentially) simplified decision table: T_D

repeat

$l \leftarrow R_C$ ▷ Keep track of the current number of rules before attempting to join them

▷ Identify joining sets associated with each rule in T_C

for i such that $1 \leq i \leq R_C$ **do**

$\beta_i \leftarrow \{r_{Ck} : |\{x_j : x_j \in X_1 \wedge t_{Ck,j} \neq t_{Ci,j}\}| \leq 1 \wedge \hat{y}_{Ck} = \hat{y}_{Ci} \wedge i \leq k \leq R_C\}$

end for

$j \leftarrow 1$

for i such that $1 \leq i \leq R_C$ **do**

▷ Only join the rule if it is the first or if it is not in previous joining sets

if $i = 1 \vee (i > 1 \wedge r_{Ci} \notin \bigcup_{k=1}^{i-1} \beta_k)$ **then**

▷ The set of variables of the new rule is the same as the 1st rule in β_i

$X_{Dj} \leftarrow X_{\beta_{i,1}}$ ▷ Denoting the 1st rule in β_i as $\beta_{i,1}$

▷ The subset of the joined rule in v is the union of the subsets in v of the rules in β_i

$d \leftarrow |X_{Ci}|$

for v such that $1 \leq v \leq d$ **do**

$t_{Dj,v} \leftarrow \bigcup_{k=i}^{R_C} \{t_{Ck,v} : r_{Ck} \in \beta_i\}$

end for

▷ Set up the joined rule's hyperrectangle

$h_{Dj} \leftarrow (t_{Dj,1}, \dots, t_{Dj,d})$

▷ The prediction of the new rule is the same as the first rule in β_i

$\hat{y}_{Dj} \leftarrow \hat{y}_{\beta_{i,1}}$

▷ The evaluation measures of the new rule are the sum of those in β_i

$Z_{Dj} \leftarrow \sum_{k=i}^{R_C} z_{Ck} [r_{Ck} \in \beta_i]$

▷ Set up the joined rule

$r_{Dj} \leftarrow (X_{Dj}, h_{Dj}, \hat{y}_{Dj}, Z_{Dj})$

$j \leftarrow j + 1$

end if

end for

▷ Set up the minimized table

$T_D \leftarrow \{r_{D1}, \dots, r_{Dj}\}$

▷ T_D is copied to T_C as the table to attempt to join rules (for the next iteration)

$T_C \leftarrow T_D$

until $l = R_C$ ▷ Repeat attempting to join rules until there is no change in the rule count

Minimizing the number of rules in the case study

Within the context of the case study, Tables 5.9 and 5.10 illustrate the process of joining rules in a decision table using the RuleGluing algorithm as a *MinimizeRuleSet* instantiation. Table 5.8 is an example. The algorithm iteratively searches for pairs of rules that share the same prediction (target variable value) and have identical conditions (subsets) for all variables except one, the joined variable. Within the example, the process goes as follows:

- Identifying joining sets – Rules r_{C1} and r_{C2} share the same prediction and subset for variable x_{14} ($t_{1,14} = t_{2,14} = [0; 1]$), implying x_1 as the joined variable. Consequently, both rules are included in the set β_1 . Similarly, rules r_{C2} and r_{C4} share the same prediction and subsets for variable x_1 ($t_{2,1} = t_{4,2} = [0; 36]$), making x_{14} the joined variable. Both rules are added to the set β_2 . Sets β_3 , β_4 , and β_5 only contain their corresponding rules (r_{C3} , r_{C4} , and r_{C5} respectively) as they do not share enough subsets with other rules. Table 5.9 presents both rules and joining sets.
- Creating rules in the simplified table – The process continues so that each joining set forms a single rule in the simplified table (Table 5.10). Joining rules involves taking the union of the subsets across all rules within the set for the joined variable. However, only rules not previously included in other sets can be joined. In β_1 , both rules can be joined to create a rule r_{D1} . In β_2 , only r_{C4} can be used as r_{C2} was already included in β_1 . Consequently $r_{D2} = r_{C4}$ and r_{C4} cannot be used again, leaving β_4 unable to create a new rule. Joining sets β_3 and β_5 each provide one rule. Once the rules to be joined are identified, their components have to be determined:
 - Prediction – Since all rules in a joining set share the same prediction, the prediction in the new rule is inherited from the first rule in the set (e.g., $\hat{y}_{D1} = \hat{y}_{C1}$).
 - Coverage – The coverage of a new rule is the sum of the coverages of all rules included in the corresponding joining set (e.g., $Coverage_{D1} = Coverage_{C1} + Coverage_{C2}$). Normalization of coverages is not required as they are either copied or joined.
- Iteration and termination – The process continues until all rules are scanned. A second attempt (Table 5.11) is performed and rules r_{C2} and r_{C4} are joined. As a third attempt does not reduce the number of rules, the final table is Table 5.12.

As the case study demonstrated, the merging of rules can occasionally introduce variables that have a neutral influence on the decision-making process. This occurs when the union operation, performed on the subsets of a merged variable, encompasses the entirety of that variable's domain. Consequently, these variables become effectively redundant, as they offer no discriminatory power within the context of their associated rules. To preserve the conciseness and efficiency of the decision table, these variables are subsequently removed with minimization. Thus, the rule

merging process can potentially reduce not only the number of rules but also the dimensionality of the decision table by eliminating variables that do not contribute to the classification task.

Table 5.9: Identifying the initial sets of rules to join (1st iteration – Table T_C)

Rule	x_{14}	x_1	$\hat{y}$	Coverage	β
r_{C1}	[0; 1]	(36; $+\infty$)	y	0.18	$\{r_{C1}, r_{C2}\}$
r_{C2}	[0; 1]	[0; 36]	y	0.36	$\{r_{C2}, r_{C4}\}$
r_{C3}	(1; $+\infty$)	(36; $+\infty$)	n	0.09	$\{r_{C3}\}$
r_{C4}	(1; 2]	[0; 36]	y	0.27	$\{r_{C4}\}$
r_{C5}	(2; $+\infty$)	[0; 36]	n	0.10	$\{r_{C5}\}$

Table 5.10: Rules joined (1st iteration – Table T_D)

Rule	Joined from set	x_{14}	x_1	$\hat{y}$	Coverage
r_{D1}	β_1	[0; 1]		y	0.54
r_{D2}	β_2	(1; 2]	[0; 36]	y	0.27
r_{D3}	β_3	(1; $+\infty$)	(36; $+\infty$)	n	0.09
r_{D4}	β_5	(2; $+\infty$)	[0; 36]	y	0.10

Table 5.11: Identifying the next sets of rules to join (2nd iteration – Table T_C)

Rule	x_{14}	x_1	$\hat{y}$	Coverage	β
r_{C1}	[0; 1]		y	0.54	$\{r_{C1}\}$
r_{C2}	(1; 2]	[0; 36]	y	0.27	$\{r_{C2}, r_{C4}\}$
r_{C3}	(1; $+\infty$)	(36; $+\infty$)	n	0.09	$\{r_{C3}\}$
r_{C4}	(2; $+\infty$)	[0; 36]	y	0.10	$\{r_{C4}\}$

Table 5.12: Rules joined (2nd iteration – Table T_D)

Rule	Joined from set	x_{14}	x_1	$\hat{y}$	Coverage
r_{D1}	β_1	[0; 1]		y	0.54
r_{D2}	β_2	(1; $+\infty$)	[0; 36]	y	0.37
r_{D3}	β_3	(1; $+\infty$)	(36; $+\infty$)	n	0.09

5.8 Specific merging scenarios

To comprehensively examine the intricacies and potential outcomes of the merging process, this section presents three additional illustrative scenarios, each involving the merging of two distinct

decision tables, denoted as T_A and T_B . In each scenarios the same instantiations of the merging functions and parameters as defined and employed in the preceding example are used, ensuring consistency in the application of the merging methodology across all scenarios. By replicating the previously established framework, the focus remains on observing the different merging results achieved when applied to diverse rule sets, rather than the influence of altering the merging parameters themselves. This approach allows for a more focused analysis of how the inherent characteristics of the tables being merged contribute to the final merged outcome.

1st scenario – no common variables. Two decision tables T_A and T_B (Tables 5.13 and 5.14) are characterized by distinct sets of variables. The rule-set resulting from the intersection of these two tables, with all combinations of pairs of rules, is presented in Table 5.15.

Table 5.13: Decision table T_A (1st scenario)

Rule	x_{13}	$\hat{y}$	Coverage
r_{A1}	$[0; 4]$	y	0.86
r_{A2}	$(4; +\infty)$	n	0.14

Table 5.14: Decision table T_B (1st scenario)

Rule	x_{14}	$\hat{y}$	Coverage
r_{B1}	$[0; 5]$	y	0.85
r_{B2}	$(5; +\infty)$	n	0.15

Table 5.15: Rule-set of the intersection of the decision tables (1st scenario)

Rule	x_{13}	x_{14}	$\hat{y}$	Coverage	σ
r_{A1-B1}	$[0; 4]$	$[0; 5]$	y	0.87	Successful
r_{A1-B2}	$[0; 4]$	$(5; +\infty)$	?	0.52	Conflict
r_{A2-B1}	$(4; +\infty)$	$[0; 5]$	?	0.49	Conflict
r_{A2-B2}	$(4; +\infty)$	$(5; +\infty)$	n	0.14	Successful

Following the execution of the merging process and subsequent conflict resolution, the resulting merged decision table (Table 5.16), incorporates variables originating from both T_A and T_B .

Table 5.16: Merged decision table (1st scenario)

Rule	x_{13}	x_{14}	$\hat{y}$	Coverage
r_{T1}	$[0; 4]$		y	0.69
r_{T2}	$(4; +\infty)$	$[0; 5]$	y	0.24
r_{T3}	$(4; +\infty)$	$(5; +\infty)$	n	0.07

2nd scenario – shared single variable. Two decision tables T_A and T_B share exactly the same variable (Tables 5.17 and 5.18). The rule-set resulting from the intersection of these two tables, with all combinations of pairs of rules, is presented in Table 5.19.

Table 5.17: Decision table T_A (2nd scenario)

Rule	x_{14}	$\hat{y}$	Coverage
r_{A1}	$[0; 1]$	y	0.86
r_{A2}	$(1; +\infty)$	n	0.14

Table 5.18: Decision table T_B (2nd scenario)

Rule	x_{14}	$\hat{y}$	Coverage
r_{B1}	$[0; 4]$	y	0.88
r_{B2}	$(4; +\infty)$	n	0.12

Table 5.19: Rule-set of the intersection of the decision tables (2nd scenario)

Rule	x_{14}	$\hat{y}$	Coverage	σ
r_{A1-B1}	$[0; 1]$	y	0.87	Successful
r_{A1-B2}	$\emptyset$	?	0.49	Unsuccessful
r_{A2-B1}	$(1; 4]$	?	0.51	Conflict
r_{A2-B2}	$(4; +\infty)$	n	0.13	Successful

The resulting merged decision table (Table 5.20) closely resembles table T_B , with the potential addition of information from T_A . This outcome suggests that when merging tables with identical variables, the resulting table often reflects the dominant characteristics of one of the original tables (e.g., larger hyperrectangles), with minor changes introduced by the other.

Table 5.20: Merged decision table (2nd scenario)

Rule	x_{14}	$\hat{y}$	Coverage
r_{T1}	$[0; 4]$	y	0.92
r_{T2}	$(4; +\infty)$	n	0.08

3rd scenario – one variable vs. no variables. A decision table T_A is characterized by a single variable, whereas decision table T_B contains no variables, as illustrated in Tables 5.21 and 5.22 respectively. Given this disparity, the intersection of these tables, encompassing all possible pairings, is presented in Table 5.23.

Table 5.21: Decision table T_A (3rd scenario)

Rule	x_{14}	$\hat{y}$	Coverage
r_{A1}	$[0; 1]$	y	0.89
r_{A2}	$(1; +\infty)$	n	0.11

Table 5.22: Decision table T_B (3rd scenario)

Rule	$\hat{y}$	Coverage
r_{B1}	y	1.00

Despite the presence of a variable in T_A , the resulting merged decision table, shown in Table 5.24, is identical to the T_B . This outcome highlights that the merging process, in cases where one table lacks any variables, may effectively default to the structure and rules of the table without

Table 5.23: Rule-set of the intersection of the decision tables (3rd scenario)

Rule	x_{14}	$\hat{y}$	Coverage	σ
r_{A1-B1}	$[0; 1]$	y	0.87	Successful
r_{A2-B1}	$(1; +\infty)$	?	0.49	Conflict

variables, demonstrating that the presence of variables in one table does not necessarily influence the final merged table in such scenarios.

Table 5.24: Merged decision table (3rd scenario)

Rule	$\hat{y}$	Coverage
r_{T1}	y	1.00

5.9 Converting a decision table to a model

In the context of merging multiple models, maintaining the interpretability of the resulting merged model is paramount. This requires that the format of the merged model conforms to the format of the base models. For example, if all input models are decision trees, the merged model must be represented as a decision tree. To ensure this consistency of representation, the abstract function *ConvertFromTable* is used. This function takes a decision table as input and returns an equivalent model in the desired format. *ConvertFromTable* functions as a complement to the *ConvertToTable* function, which performs the inverse operation, transforming a model in a specific format into its decision table representation.

To illustrate the process of deriving a decision tree from a decision table, consider the conversion process, which is theoretically straightforward. By analyzing the structure of the decision table, each rule can be directly mapped to a corresponding branch within the resultant decision tree. Under typical circumstances, this conversion between decision tables and decision trees is considered lossless, implying that no information is lost during the transformation. As a concrete example, consider Table 5.25, which, when converted, yields a decision tree (x_1 = ‘Age’ and x_{14} = ‘Delayed courses’ from Table 4.1). This example demonstrates the direct correspondence between

Table 5.25: Decision table

Rule	x_{14}	x_1	$\hat{y}$	Coverage
r_1	$[0; 1]$		y	0.54
r_2	$(1; +\infty)$	$[0; 36]$	y	0.37
r_3	$(1; +\infty)$	$(36; +\infty)$	n	0.09

Listing 5.2: Decision tree from a decision table

```

delayed_courses <= 1: y
delayed_courses > 1:
...age <= 36: y
   age > 36: n

```

the rules of the table and the branches of the resulting decision tree, highlighting the fidelity of the conversion process.

However, merging decision tables can introduce a critical challenge, namely the potential loss of completeness with respect to the decision space. This loss arises from the rejection of rules during the merging process, leading to uncovered regions within the decision space. These uncovered regions correspond to specific combinations of independent variable values for which the resultant decision table fails to provide an applicable rule, effectively leaving these regions without a prediction. The loss of completeness makes the direct derivation of a decision tree from the decision table infeasible. This is because a decision tree, by its nature, requires a defined outcome for every possible path through its branches, a requirement that an incomplete decision table cannot fulfill.

Addressing the challenge raised by the loss of completeness in specific merging scenarios requires the development of quantitative metrics. These metrics, detailed in Appendix B, are used to quantify the extent of completeness within the merged models. Furthermore, Chapter 6 explores methodologies for deriving a complete model from an incomplete rule-set, thereby providing concrete approaches for instantiating the *ConvertFromTable* function, enabling the generation of a complete model even from a potentially incomplete decision table representation.

5.10 Summary

Merging a set of rule-based models into a single model follows a framework-like structure (hence its inclusion in the *Inmplode* framework), consisting of a generic workflow and customizable processes. In the method presented in this chapter, all models are transformed into a common representation, the extended attributive decision table. This isolates the initial model representation from the merging approach. Another assumption is that a decision rule's representation includes additional elements to store information necessary for the merging process.

Models are merged pairwise and sequentially. The merging strategy is customizable and illustrated with rule intersection, a highly intuitive process. However, other rule-merging approaches can be implemented. Since there is no guarantee that a pair of decision tables can be merged, the process verifies if the resulting rule set can yield a valid model. If successful, the problem of conflicting rules, i.e., rules making different predictions for the same decision space region, must be resolved. This resolution process is also customizable.

Finally, before concluding the merging process, the decision table is simplified to reduce its size (a metric of interpretability). This process is also customizable and illustrated with a rule gluing method that iteratively attempts to combine rules and reduce the number of variables the model uses if they become redundant. However, it was identified that the decision table might be incomplete, i.e., not covering the entire decision space, raising the issue of incompleteness, which hinders its use. This issue is addressed in Chapter 6, where the goal is to derive complete models even when faced with incomplete rule sets.

Chapter 6

Methods to Complete Consensus Rule-based Models

The *Inmplode* framework, introduced in Chapter 4, develops consensus models from multiple operational models, with Chapter 5 detailing the merging process. The operational models are decision trees, which, due to their property of exhaustiveness, are complete models. As a result, they cover the entire decision space and can provide predictions for any combination of independent variables. However, a key limitation of this process is that rule removal during merging can lead to incomplete decision tables, lacking full coverage of the decision space, creating regions where no prediction can be made. To address this limitation, this chapter proposes two novel methods to complete consensus rule-based models: Synthetic Data Trees and Distributional Data Trees.

The first approach, *Synthetic data trees*, generates synthetic data to represent the regions of the decision space covered by each rule in the merged decision table. This synthetic dataset is then used to construct a decision tree using standard algorithms such as C5.0 or CART. A key challenge in this approach is to create a synthetic dataset that adequately captures the underlying data distribution. If this is not achieved, the resulting decision tree may be incomplete, failing to cover all regions of the decision space.

The second approach, *Distributional data trees*, explores the parallel between the regions defined by rules in the merged decision table and distributional data within the Symbolic Data Analysis paradigm. It treats the rule conditions as distributional symbolic variables, representing them as probability distributions that inherently cover the empirical domain of each variable. A decision tree is then constructed using an algorithm specifically designed to handle distributional data, thus ensuring the resulting model's completeness.

6.1 Training decision trees from incomplete decision tables

Two distinct approaches for training decision trees from incomplete decision tables are presented. These methods are demonstrated using the publicly available Bank Marketing dataset (Moro et al., 2014) which provides customer information from a bank with independent variables such as Age,

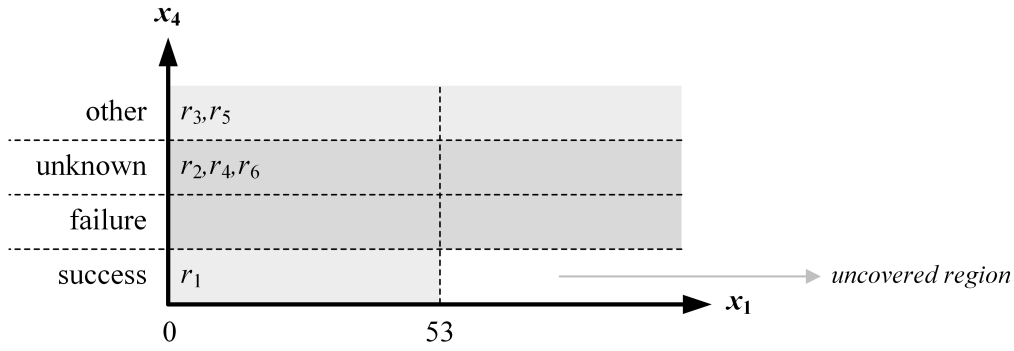


Figure 6.1: Uncovered region in a decision space

Balance, Contact (form of communication with a customer), and Poutcome (outcome of previous marketing campaigns). The target variable is Subscribe, indicating whether a customer enrolled in a marketing campaign. The analysis focuses on the first six months of the year, where decision tree models are trained for each month, using the C5.0 algorithm (Kuhn and Quinlan, 2023). Merging the six operational models results in the decision table presented in Table 6.1.

Table 6.1: Decision table resulting from merging six models (Bank marketing dataset)

Rule	Age (x_1)	Balance (x_2)	Contact (x_3)	Poutcome (x_4)	Subscribed ($\hat{y}$)
r_1	$(-\infty; 53]$			{success}	yes
r_2		$(-\infty; 2137]$		{failure, unknown}	no
r_3		$(-\infty; 2137]$		{other}	yes
r_4		$(2137; +\infty)$	{cellular, telephone}	{failure, unknown}	no
r_5		$(2137; +\infty)$	{cellular, telephone, unknown}	{other}	yes
r_6		$(2137; +\infty)$	{unknown}	{failure, unknown}	yes

Analyzing rule r_1 in the table reveals that it only has a condition related to the variable Age (x_1). This implies that r_1 is triggered for any combination of values in the other variables. Consequently, empty cells can be interpreted as representing the entire domain (all possible values) for the corresponding variables. However, a decision tree directly derived from Table 6.1 would be incomplete. To illustrate, consider the decision space defined by the variables Age (x_1) and Poutcome (x_4) depicted in Figure 6.1, with the regions covered by the rules in Table 6.1. Clearly, an example with $x_1 = 75$ and $x_4 = \text{'success'}$ falls into an uncovered region. Therefore, the table lacks a specific rule triggered by this combination (regardless of the other variables), highlighting the difficulty of directly constructing a decision tree from this incomplete table. The following sections present approaches to address the challenge of creating a decision tree from an incomplete decision table.

6.2 Synthetic data trees method

The first method to train a decision tree from an incomplete decision table is depicted in Figure 6.2, comprising two tasks. This approach focuses on generating a synthetic dataset (D_s) that represents the decision space defined by the rules in the table. To achieve this, the first task involves creating D_s , composed of single-valued data derived from the decision table (T). This generation process is guided by the distribution of variables, captured in a multivariate histogram (Γ_f). The second task uses this synthetic dataset to create a model (M), using any standard propositional decision tree algorithm. Both this method and the resulting decision trees are named *Synthetic data trees*. While this approach aligns with *Augmented Data* (Alomar et al., 2023) techniques, which extend existing datasets, the term Synthetic Data Trees highlights the creation of entirely new data through a defined generation process.

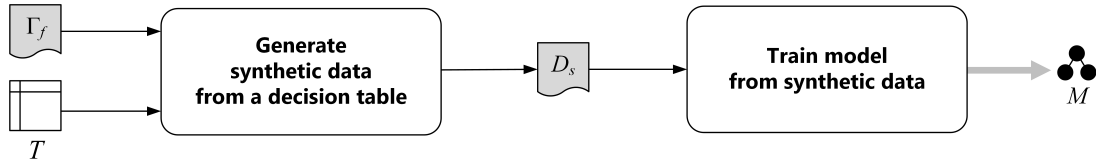


Figure 6.2: Synthetic data trees method

6.2.1 Generate synthetic data from a decision table

In the first task of the process, a synthetic dataset containing examples of single-valued data is generated. The dataset is constructed by analyzing the regions of the decision space covered by each rule, referred to as the rule's hyperrectangle. Examples are generated such that their values lie within the boundaries defined by the rule, forming a cloud of data enclosed within the hyperrectangle. Although some regions may not be covered by the rules, and therefore no data is generated for these areas, this issue is addressed in the subsequent step. A decision tree is trained, leveraging its inherent property as a complete model to ensure coverage of any previously uncovered regions of the decision space.

As introduced in Section 5.3, formally, a rule (Equation 5.1) is defined as a conjunction of propositions, each specifying conditions that the values of independent variables (x_j) must satisfy to predict a specific target variable $\{\hat{y}\}$. These propositions involve checks of value membership against subsets (t) of variable domains, which may consist of either categorical or numerical sets of values, or intervals. For convenience, the structure of a rule r_i and the subsets are revisited:

$$r_i : (x_1 \in t_{i,1}) \wedge (x_2 \in t_{i,2}) \wedge \cdots \wedge (x_d \in t_{i,d}) \longrightarrow \hat{y}_i$$

$$t_{i,j} = \{c_1, c_2, \dots, c_j\} \quad \text{or} \quad t_{i,j} = [a; b]$$

For each rule r_i in a decision table, a *synthetic rule dataset*, denoted as E_i (Equation 6.1), is assembled by performing the Cartesian product of its subsets $(t_{i,j})$, combined with a set containing only the target variable value $(\{\hat{y}_i\})$. Subsequently, all the rule datasets are joined into a single dataset D_s (Equation 6.2).

$$E_i = t_{i,1} \times t_{i,2} \times \cdots \times t_{i,d} \times \{\hat{y}_i\} \quad (6.1) \quad D_s = \bigcup_{i=1}^R E_i \quad (6.2)$$

Determining the number of values to be included in the dataset from each subset (t) is critical. A higher number of values will lead to more examples being generated. While many examples might seem beneficial, it is essential to counterbalance the potential drawback of a very large dataset. This becomes especially important when dealing with a significant number of variables (more than 10) with large domains of numerical or high cardinality categorical variables. In such cases, the dataset can rapidly grow to a volume that makes model training computationally expensive or even infeasible. Therefore, careful selection of which values of each variable should be included in the synthetic dataset is essential.

To select values from a given domain subset, a method is applied to generate a discrete set of values based on the defined subset t_i . The result is a *generated subset*, denoted as t_i^{gen} . Thus, for a given variable x , with subset t_i , a GenerateValues method can be applied to create t_j^{gen} (Equation 6.3). The generated subsets are then used to create synthetic rule datasets, replacing the original subsets within the underlying rule. Each method also includes a parameter n , which defines the number of values to be generated within the subset. This parameter should be selected according to the specific requirements of the dataset and computational constraints.

$$t_i^{\text{gen}} = \text{GenerateValues}(t_i, n) \quad (6.3)$$

The following sections describe methods for generating synthetic values from a given subset t_i of a variable x . These methods are formalized for both numerical variables, defined within the range $[a; b]$, and categorical variables, where $C = \{c_1, c_2, \dots, c_k\}$ represents the set of categories within t_i . Some methods may result in the repetition of values, meaning that t_i^{gen} can be a *multiset*, which is a set in which elements are allowed to repeat (Blizard, 1991).

6.2.1.1 Data distribution-based generation

This method generates values for numerical variables within the range $[a; b]$ by sampling according to the probabilities defined by the empirical distribution $P(x)$ of variable x restricted to the interval $[a; b]$ (Equation 6.4). For categorical variables, values are sampled based on the observed category frequencies (Equation 6.5), where $P(c_i)$ represents the probability (or relative frequency) of category c_i in t_i .

$$t_i^{\text{gen}} = \{v_1, v_2, \dots, v_n\}, \quad \text{where } v_k \sim P(x \in [a; b]) \quad (6.4)$$

$$t_i^{\text{gen}} = \{c_1, c_2, \dots, c_n\}, \quad \text{where } c_k \sim P(C) \quad (6.5)$$

6.2.1.2 Uniform distribution-based generation

This method generates values uniformly within t_i . For numerical variables, it produces evenly spaced values across the range $[a; b]$ (Equation 6.6). For categorical variables, the values are drawn from the set of categories in t_i (Equation 6.7). If $n < |C|$, the result is a subset of C , prioritizing the most frequent categories.

$$t_i^{\text{gen}} = \left\{ a + k \frac{b-a}{n-1} : k = 0, 1, \dots, n-1 \right\} \quad (6.6)$$

$$t_i^{\text{gen}} = C \quad (6.7)$$

6.2.1.3 Random distribution-based generation

This method generates values randomly within t_i . For numerical variables, values are sampled randomly from a continuous uniform distribution over the range $[a; b]$ (Equation 6.8). For categorical variables, values are sampled randomly with replacement from the set C , corresponding to a discrete uniform distribution (Equation 6.9).

$$t_i^{\text{gen}} = \{v_1, v_2, \dots, v_n\}, \quad \text{where } v_k \sim U(a, b) \quad (6.8)$$

$$t_i^{\text{gen}} = \{c_1, c_2, \dots, c_n\}, \quad \text{where } c_k \sim U(C) \quad (6.9)$$

6.2.1.4 Central tendency-based generation

This method generates n identical values based on a measure of central tendency derived from the original subset t_i . For numerical variables, the generated values can be the mean, median, or mode of the values in $[a; b]$ (Equation 6.10). For categorical variables, the generated values correspond to the mode in set C (Equation 6.11).

$$t_i^{\text{gen}} = \underbrace{\{\mu, \dots, \mu\}}_n \quad \text{or} \quad t_i^{\text{gen}} = \underbrace{\{\text{med}, \dots, \text{med}\}}_n \quad \text{or} \quad t_i^{\text{gen}} = \underbrace{\{\text{mode}, \dots, \text{mode}\}}_n \quad (6.10)$$

$$t_i^{\text{gen}} = \underbrace{\{\text{mode}(C), \dots, \text{mode}(C)\}}_n \quad (6.11)$$

6.2.1.5 Boundary-based generation

This method generates values based on the boundaries of the original subset t_i , applicable to numerical variables only. It can generate both boundaries (Equation 6.12) or only the minimum (Equation 6.13) or maximum value (Equation 6.14), repeated to obtain a multiset of size n .

$$t_i^{\text{gen}} = \{a, b\} \text{ repeated to size } n \quad (6.12)$$

$$t_i^{\text{gen}} = \underbrace{\{a, a, \dots, a\}}_n \quad (6.13)$$

$$t_i^{\text{gen}} = \underbrace{\{b, b, \dots, b\}}_n \quad (6.14)$$

Table 6.2 illustrates the application of each method for $t_{1,1} = [18; 53]$, with $n = 10$, each a different alternative for $t_{1,1}^{\text{gen}}$.

Table 6.2: Methods for generating values from a subset of a hyperrectangle

Method	Example of generated values for interval Age = [18; 53]
Data distribution-based	{25, 28, 32, 35, 38, 42, 45, 48, 50, 53}
Uniform distribution-based	{18, 21, 25, 29, 33, 37, 41, 45, 49, 53}
Random distribution-based	{19, 31, 32, 35, 42, 43, 44, 47, 48, 50}
Central tendency-based	{40, 40, 40, 40, 40, 40, 40, 40, 40, 40} (median)
Boundary-based	{18, 18, 18, 18, 18, 53, 53, 53, 53, 53}

Each rule, from r_1 to r_6 , creates a corresponding synthetic rule dataset (E_1 to E_6). Table 6.3 shows a sample of the generated synthetic dataset (D_s), using uniform distribution-based generation as method to generate values.

6.2.2 Train a model from synthetic data

The second task of the process involves training a decision tree using the generated synthetic dataset. Standard propositional decision tree algorithm appropriate for single-valued data such as C5.0 (Kuhn and Quinlan, 2023) or CART (Breiman et al., 1984) can be used for this purpose.

The synthetic data trees method does not guarantee completeness, as illustrated in the following example. Even though the synthetic dataset generation process can account for all the categories in categorical variables, the resulting decision tree might still fail to fully capture the entire domain. This is evident in the decision tree shown in Listing 6.1, trained on the synthetic dataset using the C5.0 algorithm.

The decision tree is incomplete because it does not cover all the values of the Contact variable (x_3). The domain of x_3 includes three categories: cellular, telephone, unknown. However, the decision tree only tests for ‘cellular’ and ‘unknown’. It fails to provide a prediction when the value is ‘telephone’. This limitation arises because, while decision trees for numerical variables inherently divide the domain into mutually exclusive intervals, categorical variables can lead to gaps. Even if all categories are explicitly represented during synthetic data generation, the decision

Table 6.3: A single-valued synthetic dataset (D_s)

	Age (x_1)	Balance (x_2)	Contact (x_3)	Poutcome (x_4)	Subscribed ($\hat{y}$)
E_1	18	-8019	cellular	success	yes
	$\vdots$	$\vdots$	$\vdots$	success	yes
	53	102127	unknown	success	yes
E_2	18	-8019	cellular	failure	no
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	no
	95	2137	unknown	unknown	no
E_3	18	-8019	cellular	other	yes
	$\vdots$	$\vdots$	$\vdots$	other	yes
	95	2137	unknown	other	yes
E_4	18	2138	cellular	failure	no
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	no
	95	102127	telephone	unknown	no
E_5	18	2138	cellular	other	yes
	$\vdots$	$\vdots$	$\vdots$	other	yes
	95	102127	unknown	other	yes
E_6	18	2138	unknown	failure	yes
	$\vdots$	$\vdots$	unknown	$\vdots$	yes
	95	102127	unknown	unknown	yes

tree might not account for all possible values, resulting in incomplete coverage of the variable's domain.

6.3 Distributional data trees method

The second method for training a decision tree from an incomplete table uses distributional data, a form of symbolic data. Since this is not a standard data type, the following sections introduce the Symbolic Data Analysis paradigm to clarify its relevance to the method, followed by a discussion of its parallel to decision tables.

6.3.1 The Symbolic Data Analysis paradigm

In data analysis, examples are defined by numerical or categorical variables that capture multiple attributes of the entities under investigation. Typically, these units are structured within a dataset, and conventional analytical techniques involve condensing the data using statistical measures such

Listing 6.1: Decision tree trained from synthetic data with the C5.0 algorithm

```

poutcome in {other,success}: yes
poutcome = unknown:
:...contact = cellular: no
    contact = unknown:
        :...balance <= 0: no
            balance > 0: yes

```

as means, medians, or modes. Nonetheless, these approaches frequently encompass simplifying the data, which can lead to a certain level of information loss (Brito, 2014).

The *Symbolic Data Analysis* (SDA) paradigm (Bock and Diday, 2000; Diday, 2016) offers a different perspective, introducing the concept of *symbolic variables* as a means of aggregating statistical units. Unlike traditional methods that condense individual entity information into a single numerical value, SDA aims to preserve the intrinsic variation in the data (Diday, 2016). The transition from single-valued data to symbolic data marks a significant shift in how statistical units are represented.

To clarify the distinction, traditional variables are referred to as *single-valued variables*. In contrast, variables derived through SDA are *symbolic variables*, which aggregate values to capture the inherent variability within the data. Rather than representing data points with single values, symbolic variables encapsulate *symbolic values*. These symbolic values can take several forms, such as sets, intervals, or distributions that span a specific domain. For instance, instead of reducing a collection of data points to a mean value, the SDA approach represents it as a set of values, an interval capturing the range, or a distribution reflecting the frequency of different values within the data.

Figure 6.3 provides a concise overview of the taxonomy of SDA variable types and their mapping to single-valued variables. Single-valued variables include traditional numerical (discrete or continuous) and categorical variables. In this context, numerical variables are referred to as *quantitative single-valued variables*, containing single-value data. When transitioning to symbolic variables, aggregation can result in either a *quantitative multi-valued variable*, a set of individual values, or an *interval-valued variable*, representing the variable as an interval of values, capturing the minimum and maximum values. For categorical variables, the transition to symbolic variables always results in a set of values, leading to a *categorical multi-valued variable*. All these cases are collectively known as holding *set-valued data*.

A more enriched representation involves incorporating the frequency distribution of values. For both quantitative multi-valued and interval-valued variables, this leads to a *histogram-valued variable*. For categorical multi-valued variables, the representation of values and frequencies is referred to as a *categorical modal variable*. Symbolic values that include frequencies are referred to as *distributional data*. This chapter presents examples of all these variable types in appropriate contexts.

Table 6.4 presents a sample dataset of student enrollments in a university. Each student is an

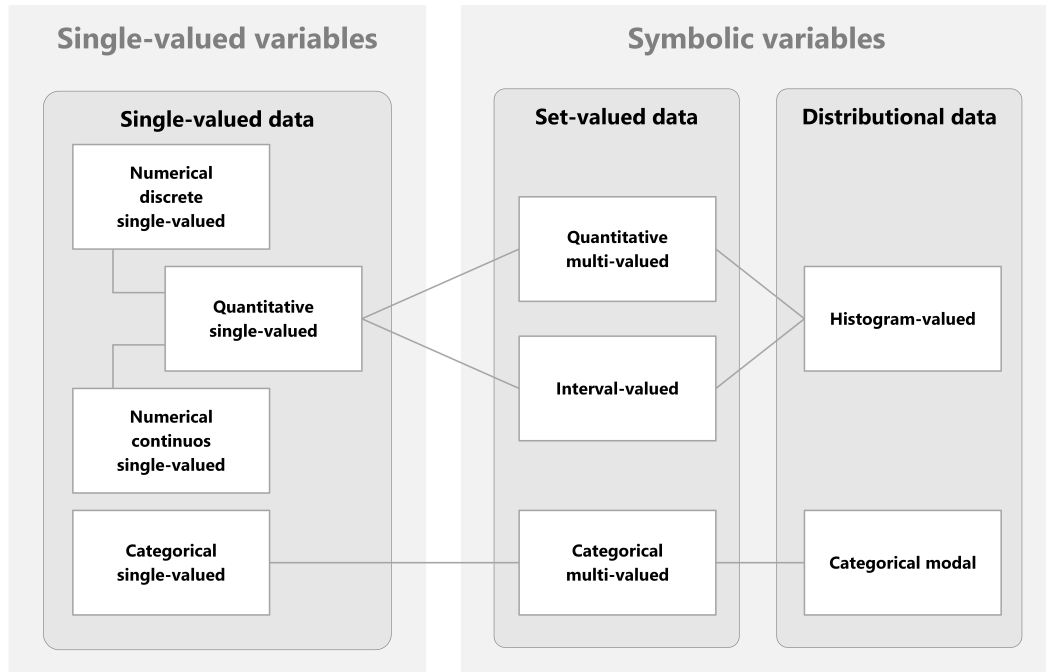


Figure 6.3: Taxonomy of single-valued and symbolic variables

example described by single-valued variables. The dataset is characterized by the single-valued variables listed in Table 6.5. Table 6.6 presents a summarized view of the data in a *symbolic data table*, where the information is aggregated by course. To distinguish between single-valued and corresponding symbolic variables, the latter is represented by an apostrophe. For instance, the numerical discrete single-valued variable (x_3) and the interval-valued variable (x'_3), both relating to age. Another example is the categorical single-valued variable (x_4) and the categorical multi-valued variable (x'_4), both relating to gender.

Like single-valued variables, symbolic variables are also defined by their associated domains. Generally, the *domain* of a variable, whether single-valued or symbolic, is the set of potential values it can hold. When considering data derived from a dataset, this concept maps to the *empirical domain*, denoted by $\hat{\text{dom}}$. This empirical domain comprises all the distinct values observed in the examples within that specific dataset (Brito, 2014). Consequently, the set $\hat{\text{dom}}.x$ encapsulates the range of possible values for a given variable x .

The concept of a domain takes on a new meaning when dealing with symbolic variables. Unlike single-valued variables with a domain encompassing a set of values, the domain of a symbolic

Table 6.4: Student enrollments data

Student ID (x_1)	Course (x_2)	Age (x_3)	Gender (x_4)	Marital status (x_5)	Distance (x_6)
1	Econometrics	19	Female	Single	2
2	Geometry	20	Male	Single	1
3	Anatomy	35	Male	Married	4

Table 6.5: Student enrollments variables

Variable	Description	Type
$x_1 = \text{Student \#}$	Unique student identification number	Numerical discrete
$x_2 = \text{Course}$	Student's enrolled course	Categorical
$x_3 = \text{Age}$	Age of student at enrollment date	Numerical discrete
$x_4 = \text{Gender}$	Male, Female	Categorical
$x_5 = \text{Marital status}$	Single, Married, Divorced, ...	Categorical
$x_6 = \text{Distance}$	Distance from campus to home (km)	Numerical continuous

Table 6.6: Course enrollments data

Course (x_2)	Age (x'_3)	Gender (x'_4)	Marital status (x'_5)	Distance (x'_6)
Econometrics	[18;35]	{Female, Male}	{Single, Married}	[1;20]
Geometry	[19;29]	{Female, Male}	{Single, Married}	[1;12]
Anatomy	[18;19]	{Female, Male}	{Single}	[1;4]

Table 6.7: Course enrollments variables

Variable	Description	Type
$x_2 = \text{Course}$	Student's enrolled course	Categorical single-valued
$x'_3 = \text{Age}$	Age range of students at enrollment date	Interval-valued
$x'_4 = \text{Gender}$	Set of genders	Categorical multi-valued
$x'_5 = \text{Marital status}$	Set of marital status	Categorical multi-valued
$x'_6 = \text{Distance}$	Distance from campus to home range (km)	Interval-valued

variable, denoted as $\text{dom } x'$, encompasses sets of elements.

For quantitative multi-valued variables and categorical multi-valued variables, the domain of x' is derived from the *power set* of the domain of x . A power set, denoted by $\mathcal{P}$, encompasses all possible subsets of a set, including both the empty set and the set itself (Jech, 2006). Consider a set $S = \{a, b, c\}$ as an example to clarify this concept. The power set of S , denoted by $\mathcal{P}(S)$, is composed of the subsets $\{\}$ (the empty set), $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, and $\{a, b, c\}$. In the case of quantitative multi-valued variables and categorical multi-valued variables $\text{dom } x'$ is the power set of $\hat{\text{dom}} x$ except for the empty set (Equation 6.15). For interval-valued variables, $\text{dom } x'$ is defined as the collection of all possible intervals where the lower and upper bounds of these intervals are integer or real values that are part of the single-valued variable domain $\hat{\text{dom}} x$ (Equation 6.16). Furthermore, these intervals must be mutually exclusive and non-overlapping.

$$\text{dom } x' = \mathcal{P}(\hat{\text{dom}} x) - \{\}$$
(6.15)

$$\text{dom } x' = \{[l; u] : l, u \in \hat{\text{dom}} x \wedge u > l\} \quad (6.16)$$

In recent years, the field of Symbolic Data Analysis has seen significant advancements, particularly in methods for analyzing interval data, which have received the most attention. In classification, decision trees (Bock and Diday, 2000) and k -nearest neighbor methods (Appice et al., 2006) have been adapted for symbolic data, while other approaches utilize imprecise probabilities and advanced techniques like Support Vector Machines and Bayesian decision trees (Angulo et al., 2007; Ak et al., 2015).

Among multivariate methods, cluster analysis has been a key focus (Brito, 2014). SDA also encompasses extensions to factorial analysis (e.g., adaptations of PCA using midpoints, ranges, or quantile functions (Cazes et al., 1997; Diday, 2011)), multiple regression for interval data (Dias and Brito, 2013), and even time series analysis with models like Vector Autoregressive (García-Ascanio and Maté, 2010) and applications in finance (González-Rivera and Arroyo, 2012).

6.3.2 Decision tables as symbolic data

Symbolic data analysis represents data by sets, intervals, or distributions. Decision tables, on the other hand, are rule-based structures. Each rule in a decision table represents a condition, expressed as a combination of propositions, that predicts a value for a target variable. All propositions within a decision table use set membership. Consequently, decision tables inherently hold set-valued data, often represented within quantitative multi-valued or interval-valued variables.

This characteristic of decision tables aligns with the concept of symbolic data tables. Since the rules in a decision table express conditions based on set membership, all variables are inherently symbolic. Unlike traditional data analysis workflows, variable aggregation is not directly applicable, as the symbolic values themselves represent the data's inherent characteristics. This parallel between decision tables and symbolic data forms the basis for the second method of training a decision tree from an incomplete table. To ensure a complete model, this approach relies on using distributional data, which includes information from all values across all variables, and trains a decision tree based on this information.

This process comprises two tasks, as depicted in Figure 6.4. First, the decision table rules (T) are transformed into distributional data (T_d). This transformation uses a multivariate histogram (Γ_f) to represent the distribution of variables. Subsequently, a decision tree model (M) is trained using the TREE algorithm, specifically designed to handle distributional data (Brito, 2014; Périnel and Lechevallier, 2000). Both this method and the resulting decision trees are termed *Distributional Data Trees*, reflecting the use of distributional data during the training process.

6.3.3 Generate distributional data from a decision table

The first task involves transforming the set-valued symbolic values in the decision rules into distributional data. This necessary step, which uses existing techniques (Brito, 2014), requires the

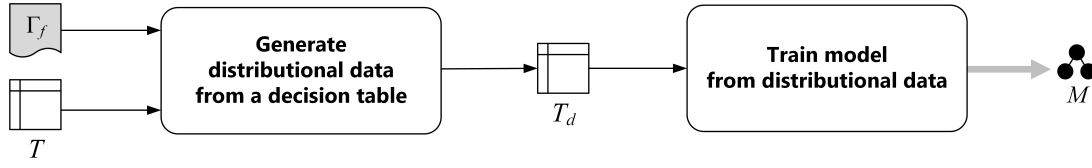


Figure 6.4: Distributional data trees method

variable distribution, which can be extracted from a histogram-based density estimator. The following sections provide details and examples of how categorical multi-valued and interval-valued variables are transformed.

6.3.3.1 Categorical multi-valued variables

Categorical multi-valued variables hold multiple discrete values and are transformed into categorical modal variables (Brito, 2014). Considering the subset $t = \{v_1, v_2, \dots\}$, the conversion consists of the following steps:

1. Normalizing relative frequencies – Assuming the frequency of each value has previously been calculated, this step involves calculating the normalized relative frequency (f_k'') for each value (v_k) within the set t (Equation 6.17). This ensures that the sum of all normalized frequencies equals one, reflecting a valid probability distribution.

$$f_k'' = \frac{f_k}{\sum_{v_i \in t} f_i} \quad (6.17)$$

2. Generating distributional data – The symbolic value (t') for a categorical modal variable is represented as a set of pairs, where each pair holds the original value (v_k) with its corresponding normalized relative frequency (f_k'') (Equation 6.18). In this distributional data format, each pair element is separated by a colon (':'), and the normalized relative frequency is enclosed in parentheses '()' for clarity.

$$t' = \{v_k (f_k'') : v_k \in t\} \quad (6.18)$$

The process is illustrated with the conversion process for a categorical multi-valued variable into a categorical modal variable, specifically, the symbolic value for the Contact (x_3) variable, in the fourth row of Table 6.1, into a distributional representation (as $v_3 \notin t_{4,3}, f_3'' = 0$):

$$t_{4,3} = \{\text{cellular}, \text{telephone}\} \longrightarrow t'_{4,3} = \{\text{cellular} (f_1''), \text{telephone} (f_2''), \text{unknown} (0)\}$$

The process involves calculating the normalized frequency for each value in $t'_{4,3}$. The calculation of frequencies relies on the distribution of the Contact variable (Table 6.8).

Table 6.8: Distribution of Contact in the Bank marketing Dataset

k	v_k	f_k
1	cellular	0.64774
2	telephone	0.06428
3	unknown	0.28798

$$f_1'' = \frac{f_1}{f_1 + f_2} = 0.90972$$

$$f_2'' = \frac{f_2}{f_1 + f_2} = 0.09028$$

The resulting symbolic value representing distributional data is given by:

$$t'_{4,3} = \{\text{cellular} (0.90972), \text{telephone} (0.09028), \text{unknown} (0.00000)\}$$

6.3.3.2 Interval-valued variables

Interval-valued variables represent a continuous range of values and are transformed into histogram-valued variables (Brito, 2014). Considering the interval $t = [l_t; u_t]$, the conversion consists of the following steps:

1. Selecting relevant bins – Defining a subset $\mathcal{B}'$ of the original bin set $\mathcal{B}$. This subset includes only the bins in the symbolic value $t = [l_t; u_t]$. A bin $b_k = [l_{b_k}; u_{b_k}]$ is selected if entirely contained within t or it includes either the lower limit or the upper limit (Equation 6.19). If the limit of t is negative or positive infinity, it should be replaced by the lowest or highest limit of the distribution, respectively.

$$\mathcal{B}' = \{b_k : (b_k \subseteq t) \vee (l_t \in b_k) \vee (u_t \in b_k)\} \quad (6.19)$$

2. Adjusting relative frequencies – Assuming the frequency of each value has previously been calculated, this step involves adjusting these values to account for the selected bins in $\mathcal{B}'$. If a bin includes the upper limit of the original value set (t), its adjusted relative frequency (f'_k) represents the proportion of data points that fall within the intersection between the bin and the interval t . Otherwise, the adjusted relative frequency for the bin remains its original frequency (Equation 6.20).

$$f'_k = \begin{cases} f_k \frac{u_t - l_{b_k}}{u_{b_k} - l_{b_k}} & \text{if } u_t \in b_k, \\ f_k & \text{if } l_t \notin b_k \end{cases} \quad (6.20)$$

3. Normalizing relative frequencies – As with categorical variables, this step involves calculating the normalized relative frequency (f''_k) for each selected bin (b_k) in $\mathcal{B}'$ (Equation 6.21).

$$f''_k = \frac{f'_k}{\sum_{b_j \in \mathcal{B}'} f'_j} \quad (6.21)$$

4. Generating distributional data – The symbolic value (t') for a histogram-valued variable is a set of pairs, where each pair holds each selected bin (b_k) with its corresponding normalized relative frequency (f_k'') (Equation 6.22).

$$t' = \{b_k(f_k'') : b_k \in \mathcal{B}'\} \quad (6.22)$$

The process is illustrated with the conversion process for an interval-valued variable into a histogram-valued variable, specifically, the symbolic value for the Age variable (x_1), in the first row of Table 6.1, into a distributional representation (containing m bins):

$$t_{1,1} = (-\infty; 53] \longrightarrow t'_{1,1} = \{b_1(f_1''), \dots, b_m(f_m'')\}$$

The process begins by identifying the bins of the frequency distribution (Table 6.9) covered by this range. The lower limit is negative infinity ($-\infty$), so the distribution encompasses all data points from the first bin onwards. The upper limit (53) restricts the distribution to the fifth bin (50;60]. Therefore, the set of all bins covered by the symbolic value are:

$$\mathcal{B}' = \{b_1, b_2, b_3, b_4, b_5\}$$

Following this identification, the distributional representation is comprised of pairs of bins and corresponding frequencies ($f_k'' = 0, b_k \notin \mathcal{B}$):

$$t'_{1,1} = \{b_1(f_1''), b_2(f_2''), b_3(f_3''), b_4(f_4''), b_5(f_5''), b_6(0), b_7(0), b_8(0), b_9(0)\}$$

Table 6.9: Distribution of Age in the Bank marketing Dataset

k	b_k	f_k
1	(10;20]	0.00215
2	(20;30]	0.15335
3	(30;40]	0.39121
4	(40;50]	0.24859
5	(50;60]	0.17843
6	(60;70]	0.01551
7	(70;80]	0.00858
8	(80;90]	0.00203
9	(90;100]	0.00015

Calculating adjusted frequencies relies again on the Age variable's distribution. All bins except b_5 fall entirely within the symbolic value range ($-\infty; 53]$ and inherit their frequencies directly from the distribution. However, the fifth bin ($b_5 = (50; 60]$) partially overlaps with the range ($(50; 53]$).

Therefore, the frequency assigned to this bin is adjusted to reflect only the portion within the specified age range.

$$f'_5 = f_5 \frac{u_t - l_{b_5}}{u_{b_5} - l_{b_5}} = 0.17843 \frac{53 - 50}{60 - 50} = 0.05353$$

Following the same normalization procedure as the previous example, the adjusted frequencies (f'_i) are divided by their sum to obtain the normalized relative frequencies (f''_i). The resulting symbolic value representing distributional data is given by:

$$t'_{1,1} = \{(10;20] (0.00253), (20;30] (0.18067), (30;40] (0.46088), \\ (40;50] (0.29286), (50;60] (0.06306), (60;70] (0.00000), \\ (70;80] (0.00000), (80;90] (0.00000), (90;100] (0.00000)\}$$

6.3.4 Train a model from distributional data

The second task of the process involves using the TREE algorithm (Périnel and Lechevallier, 2000), which is specifically designed to train a decision tree model from distributional data. Unlike standard propositional decision trees, which rely on tests—conditions that evaluate whether a given variable satisfies a specific threshold or belongs to a category—this algorithm handles symbolic data by generating conditions (referred to as questions in TREE's algorithm) that define the splits at each node of the tree. These questions guide the splitting of nodes based on the domain of each symbolic variable. The process can be subsequently divided into split generation to generate question and split evaluation, to select the ones included in the decision tree, described in the following sections.

6.3.4.1 Generation of splits

The format of the questions depends on the type of symbolic variable being considered:

- Categorical modal variables – Questions take the form of set membership, $x \in \mathcal{C}$, where x is a value, and $\mathcal{C}$ is a category subset of its empirical domain ($\mathcal{C} \subset \hat{\text{dom}} x$). Therefore, the algorithm generates questions considering all possible subsets (excluding the empty set) of $\hat{\text{dom}} x$. The total number of questions generated is $2^n - 1$.

For example, for the Contact variable (x_3) with empirical domain $\hat{\text{dom}} x_3 = \{\text{cellular}, \text{telephone}, \text{unknown}\}$, a total of 7 ($2^3 - 1$) questions are generated, as follows:

$$[x_3 \in \{\text{cellular}\}], [x_3 \in \{\text{telephone}\}], [x_3 \in \{\text{unknown}\}], \\ [x_3 \in \{\text{cellular}, \text{telephone}\}], [x_3 \in \{\text{cellular}, \text{unknown}\}], [x_3 \in \{\text{telephone}, \text{unknown}\}], \\ [x_3 \in \{\text{cellular}, \text{telephone}, \text{unknown}\}]$$

- Histogram-valued variables – Questions take the form $x \leq c$, where x represents a value and c is a specific threshold value within the empirical domain ($c \in \hat{\text{dom}}x$). The cutting threshold differs based on the type of symbolic variable before being converted to distributional data:

- Quantitative multi-valued variable – The algorithm generates $n - 1$ questions by considering each possible value in the empirical domain as the cutting threshold c .

While the variable Age (x_1) is interval-valued in the original example, it can also be treated as a quantitative multi-valued variable, as ages are discrete numbers. With 100 values in the empirical domain ($\hat{\text{dom}}x_1 = \{18, 19, \dots, 100\}$), a total of 99 ($100 - 1$) questions are generated, as follows:

$$[x_1 \leq 18], [x_1 \leq 19], [x_1 \leq 20], [x_1 \leq 21], [x_1 \leq 22], \dots, [x_1 \leq 99]$$

- Interval-valued variable – If an interval-valued variable is derived from a discrete single-valued variable (e.g., integer values), it is equivalent to the quantitative multi-valued variable case (where each value is a threshold). However, if the interval-valued variable originates from a continuous single-valued variable, there are infinitely many potential thresholds within each interval. In this case, the algorithm selects the cutting threshold as the midpoint between consecutive interval limits observed in the data (lower and upper limits). This results in $2n - 1$ possible questions, where n represents the number of intervals corresponding to each bin in the distribution.

For instance, considering the Age variable again, now treated as an interval-valued variable based on its distribution in Table 6.9, with 10 intervals in the distribution, a total of 19 ($2 \times 10 - 1$) questions are generated, as follows:

$$[x_1 \leq 10], [x_1 \leq 15], [x_1 \leq 20], [x_1 \leq 25], [x_1 \leq 30], \dots, [x_1 \leq 100]$$

6.3.4.2 Evaluation of the quality of splits

Once all questions are generated, the TREE algorithm (Périnel and Lechevallier, 2000) determines the likelihood of each question being satisfied by a given rule. This assessment is based on the information within the distributional data. Formally, for a rule (r_k) and question (q_m), the algorithm calculates the probability ($p_{k,m}$) that the rule satisfies the question. These question-probability pairs are laid out in a *probabilistic data table*, as presented in Table 6.10. This table is the basis for training a decision tree using a *recursive tree partitioning* method, a standard approach in decision tree algorithms, and used by the TREE algorithm.

The specific process for calculating probabilities depends on the type of symbolic variable containing distributional data, as follows:

- Categorical modal variables – For binary questions in the form $[x \in \mathcal{C}]$ the probability is calculated by adding the normalized frequencies of all values within the category set $\mathcal{C}$ (Equation 6.23).

Table 6.10: A probabilistic data table

Rule	q_1	$\dots$	q_m	$\dots$	q_Q
r_1	$p_{1,1}$	$\dots$	$p_{1,m}$	$\dots$	$p_{1,Q}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_k	$p_{k,1}$	$\dots$	$p_{k,m}$	$\dots$	$p_{k,Q}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r_R	$p_{R,1}$	$\dots$	$p_{R,m}$	$\dots$	$p_{R,Q}$

$$p_{k,m} = \sum_{v_j \in C} f''_{k,j} \quad (6.23)$$

- Histogram-valued variables converted from quantitative multi-valued variables – For binary questions in the form $[x \leq c]$, the probability is calculated by adding the frequencies of all values in the histogram that are less than or equal to the cutting threshold c (Equation 6.24).

$$p_{k,m} = \sum_{v_j \leq c} f''_{k,j} \quad (6.24)$$

- Histogram-valued variables converted from quantitative interval-valued variables – For binary questions of the form $[x \leq c]$, the probability is calculated by adding the frequencies of all values in the histogram that are less than the cutting threshold c added to the adjusted probability for the bin containing the threshold, accounting only for the proportion that is below c (Equation 6.25).

$$p_{k,m} = \sum_{l_{B_k} < c} f''_{k,j} + \frac{c - l_{B_k}}{u_{B_k} - l_{B_k}} f''_{k,j} \quad (6.25)$$

To demonstrate how the distributional data of variables is integrated into the probability calculations, the next two examples follow from the previously calculated symbolic values for variables Contact and Age. Considering the previous example of the categorical modal variable Contact (x'_3) with symbolic value:

$$t'_{4,3} = \{\text{cellular} (0.90972), \text{telephone} (0.09028), \text{unknown} (0.00000)\}$$

The frequencies of each category are used to calculate the probability of each question regarding the categories satisfied by the rule, as presented in Table 6.11. Considering the previous example of the histogram-valued variable Age (x'_1) with symbolic value:

Table 6.11: Probabilities for questions within $t'_{4,3}$

Probability	Question	Value
$p_{4,44}$	$q_{44} = [x_3 \in \{\text{cellular}\}]$	0.90972
$p_{4,45}$	$q_{45} = [x_3 \in \{\text{telephone}\}]$	0.09028
$p_{4,46}$	$q_{46} = [x_3 \in \{\text{unknown}\}]$	0.00000
$p_{4,47}$	$q_{47} = [x_3 \in \{\text{cellular,telephone}\}]$	1.00000
$p_{4,48}$	$q_{48} = [x_3 \in \{\text{cellular,unknown}\}]$	0.90972
$p_{4,49}$	$q_{49} = [x_3 \in \{\text{telephone,unknown}\}]$	0.09028
$p_{4,50}$	$q_{50} = [x_3 \in \{\text{cellular,telephone,unknown}\}]$	1.00000

$$t'_{1,1} = \{(10; 20] (0.00253), (20; 30] (0.18067), (30; 40] (0.46088), \\ (40; 50] (0.29286), (50; 60] (0.06306), (60; 70] (0.00000), \\ (70; 80] (0.00000), (80; 90] (0.00000), (90; 100] (0.00000)\}$$

The frequencies of each bin are used to calculate the probability of each question regarding the age being satisfied by the rule, as presented in Table 6.12.

Table 6.12: Probabilities for questions within $t'_{1,1}$

Probability	Question	Value
$p_{1,1}$	$q_1 = [x_1 \leq 10]$	0.00000
$p_{1,2}$	$q_2 = [x_1 \leq 15]$	0.00126
$p_{1,3}$	$q_3 = [x_1 \leq 20]$	0.00253
$p_{1,4}$	$q_4 = [x_1 \leq 25]$	0.09286
$p_{1,5}$	$q_5 = [x_1 \leq 30]$	0.18067
$\vdots$	$\vdots$	$\vdots$
$p_{1,19}$	$q_{19} = [x_1 \leq 100]$	1.00000

The process to create a decision tree begins by selecting the best binary split for the tree at each step. This is done by evaluating the current terminal nodes (initially, there is a single node), and for each node, examining potential splits. The criterion for selecting the best split is based in maximizing the increase in information, which measures how well the split separates different classes based on the distributional data. This information gain is calculated using a log-likelihood criterion, which accounts for the probability distribution of the classes within the current nodes.

After selecting a question, the node is split into two new terminal nodes, and the process continues recursively. If a terminal node is too small or pure (i.e., it contains only examples of one class), it is no longer split. The splitting process is repeated recursively until the tree satisfies

Listing 6.2: Decision tree trained from distributional data with the TREE algorithm

```
poutcome in {other,success}: yes
poutcome in {failure,unknown}:
:...contact in {cellular,telephone}: no
    contact = unknown: yes
```

stopping conditions, such as the size of the tree reaching a predefined maximum or no further splits being possible.

Listing 6.2 illustrates the decision tree trained using the TREE algorithm, following the transformation of the decision table in Table 6.1 into distributional data, thereby producing a complete model. A key advantage of decision trees built from distributional data is their ability to achieve completeness for both numerical and categorical variables. Traditional decision tree algorithms, such as CART (Breiman et al., 1984) and C5.0 (Kuhn and Quinlan, 2023), rely on splitting based solely on observed values within the training set. In contrast, the TREE algorithm uses the information on distributional data to account for the domain including values not present in the training set, enabling the extraction of rules that generalize effectively to unseen cases. This approach ensures that the resulting decision trees are complete, overcoming a significant limitation of standard methods.

6.4 Limitations in decision tree generation from decision tables

One key challenge in both the Synthetic data trees and Distributional data trees methods is that the successful construction of a decision tree from transformed rules is not always guaranteed. Even after transforming rules into synthetic or distributional data, a tree-building algorithm may fail to generate a tree. This limitation typically arises when the decision table lacks sufficient information to create meaningful splits. For instance, if the table contains only a single rule that cannot produce a complete model, or if the target variable remains constant regardless of input, the data lacks the necessary variability for tree construction.

When the decision table provides enough information for tree construction, the next challenge lies in the algorithm's ability to identify meaningful partitions in the data. This difficulty arises when the relationship between input variables and the target variable is weak, when noise or errors obscure underlying patterns, or when redundant input variables offer little useful information. Tree-building algorithms, such as C5.0 (Quinlan, 1986), which relies on *information gain*, and CART (Breiman et al., 1984), which uses the *Gini index*, depend on these partitions to construct trees. Similarly, the TREE algorithm (Périnel and Lechevallier, 2000) includes a parameter to define the minimum acceptable split quality. If no splits meet the criterion, the algorithm fails to generate a tree.

Another significant challenge occurs when transforming merged decision tables into decision trees. The merging process often generates many rules, resulting in high fragmentation within the

decision space, which can lead to consensus models of considerable size, sometimes comprising hundreds or even thousands of rules. If fragmentation does not occur and the number of rules remains manageable, the process of transforming the rule set into data representations for tree construction may still encounter substantial computational constraints due to the large amount of data generated. Since many rules exhibit zero or near-zero geometrical completeness, indicating limited representation in the decision space, a strategy was implemented whereby only the 20 rules with the highest geometrical completeness were selected for transformation into synthetic or distributional data. While a fixed threshold was used in the evaluation, this could be configured as an adjustable parameter within the *Inmplode* framework.

6.5 Summary

Consensus models are frequently inherently incomplete due to their development process, which limits their ability to provide predictions in uncovered regions of the decision space. Restoring full coverage requires incorporating information about the data distribution. This chapter explores two methods for addressing this issue by generating decision trees from incomplete decision rule sets that represent consensus models. While the second method ensures completeness, the first method may not guarantee it.

The first method involves generating a synthetic dataset based on the rules in the consensus model decision table and training a decision tree using established algorithms such as C5.0 or CART. While this approach can yield improved coverage compared to the original rules, it cannot guarantee completeness for categorical variables. This limitation arises because traditional decision tree algorithms rely on the observed categories in the training data, leaving gaps in the decision space for categories that are unseen or poorly represented.

The second method takes a fundamentally different approach by transforming the rules into distributional data and employing the TREE algorithm to construct decision trees. Unlike the first method, this algorithm incorporates the full domain of all variables, including categorical variables, as represented in the distributional data. By accounting for the entire range of potential categories—even those not explicitly observed in the training set—the resulting decision trees ensure completeness, fully covering the decision space.

It is paramount to evaluate the resulting models in their various forms of representation. This includes the merged decision tables, as described in Chapter 5, as well as the decision trees developed from each of the methods presented in this chapter. This evaluation encompasses a range of criteria which is the focus of Chapter 7.

Chapter 7

Empirical Evaluation of *Inmplode*

This chapter presents a comprehensive empirical evaluation of the *Inmplode* framework. Building on the theoretical foundation and methodological details established in previous chapters, the study assesses the representativeness of consensus models across multiple criteria. The analysis begins with the evaluation of the predictive performance of consensus models using traditional predictive performance metrics. This is followed by a comparative evaluation with operational models, which is then translated into a measure of potential performance loss relative to these models. Fidelity is evaluated by comparing the predictions of consensus models with those of the operational models they represent, assessing how well they capture the underlying knowledge. Traditional predictive performance metrics are also used for this evaluation. Interpretability is assessed using complexity metrics to evaluate model readability from a human perspective, complemented by a comparative analysis with operational models. Finally, completeness evaluates the decision space coverage of consensus models using completeness metrics.

The evaluation distinguishes between three representations of the consensus models: Merged decision tables, reflecting the outcome of the merging process prior to their representation as trees; and as decision trees, specifically, Synthetic data trees and Distributional data trees. The analysis reveals the strengths and limitations of each representation method, contributing to a deeper understanding of their applicability in various contexts.

7.1 Experimental setup

7.1.1 Selected datasets

The empirical evaluation of the *Inmplode* framework uses five datasets, each selected for its distinct properties and business domains. Four of these datasets are publicly accessible on Kaggle ([Kaggle, 2024](#)), specifically: Adult Census, Bank Marketing, Hotel Reservations, and Flight Delays. The fifth dataset, originates from a case study at the University of Porto, remains confidential due to data privacy constraints.

Table 7.1 summarizes each dataset, specifying the number of examples (n), the number of independent variables (d), the nature of the examples, the business units associated with the operational models, and the strata used to define the consensus models. A comprehensive description of these datasets and their stratification is provided in Appendix C.

Operational models were trained on the Adult Census dataset to predict income levels for each country of origin, and then stratified by major world regions. For the Bank Marketing dataset, which includes bank campaigns, operational models predict monthly subscription rates and were stratified by semester and quarter. In the Hotel Reservations dataset, operational models predict booking cancellations for each arrival month over two years, and then stratified by semester and quarter. The Flight Delays dataset involved predicting delays in US airport flights, with operational models relating each airport. These were then stratified by major US regions and US states. Finally, the University of Porto dataset used student enrollments to predict course failures. Operational models were trained for each course and then stratified by the faculties and the scientific areas of the courses. Except for the Adult Census dataset, two stratification alternatives were tested for each dataset. However, to maintain conciseness without loss of generality, the analysis in this chapter focuses on a single stratification criterion per dataset.

Table 7.1: Datasets used to evaluate consensus models developed with *Inmplode*

Dataset	n	d	Examples	Business units	Strata
Adult Census	49K	12	People	Countries	World regions
Bank Marketing	45K	15	Clients	Months	Semesters, Quarters
Hotel Reservations	36K	18	Reservations	Months	Semesters, Quarters
Flight Delays	539K	8	Flights	Airports	US regions, US states
University of Porto	1300K	26	Enrollments	Courses	Faculties, Scientific areas

7.1.2 Evaluation methodology

The *Inmplode* framework was deployed by training decision trees as operational models, using data specific to each business unit within the datasets. The decision trees were trained with the C5.0 algorithm (Quinlan, 1986). To ensure robust and reliable performance estimates, 10-fold cross-validation was applied, with each model being trained three times and the one with the best predictive performance selected, using the κ metric (Equation 2.16). Consensus models were then developed for each stratum, based on the configuration parameters specified in Table 7.2. Stratification variables were excluded from the training data, as they remain constant within each business unit.

The evaluation of consensus models focuses on their comparison across three representations:

- Merged table – The decision table resulting from the merging of operational models before training a decision tree (Section 5.5);

Table 7.2: *Inmplode* configuration parameters in experiments

Parameters	Settings
Datasets (D)	Dataset specific – Table 7.3
Independent variables (X)	
Target variable (y)	
Positive class ($\oplus$)	
Stratification variables (V)	
Folds (K)	10
Learning algorithm ($\mathcal{L}$)	Decision trees (C5.0)
Model evaluation ($\mathcal{Z}$)	Table 7.4
Rule evaluation metrics (Φ_r)	{Coverage, FDR, WFDR}
Conversion methods ($\mathcal{C}$)	
<i>ConvertToTable</i>	ConvertTreeTable
<i>ConvertFromTable</i>	{SyntheticDataTrees, DistributionalDataTrees}
Merging methods ($\mathcal{F}$)	
<i>MergePairRules</i>	IntersectRules
<i>EstimateRuleMeasures</i>	AverageMeasures
<i>FilterRule</i>	RejectBelowCoverage
<i>ResolveRuleConflict</i>	RuleHighestCoverage
<i>MinimizeTable</i>	RuleGluing

- Synthetic data tree – The decision tree trained on the merged table, using synthetic data derived from the rules (Section 6.2);
- Distributional data tree – The decision tree trained on the merged table, using distributional data derived from the rules (Section 6.3).

To analyze the consensus models across these representations, the evaluation is structured as follows. The results are presented separately for each dataset, detailing selected metrics for each criterion. These results summarize the average metrics across 10 experimental setups, accompanied by bar charts that highlight the most significant variations between representations for each metric, comparing consensus model representations both within and across datasets. Finally, a correlation analysis explores potential interdependencies among the metrics. More detailed results are deferred to Appendix C, which provides a complete view of metric distributions from two perspectives: strata-focused, highlighting variations between setups within each stratum, and setup-focused, emphasizing variations across strata within each setup. While these detailed results provide context, they are not critical for addressing the hypothesis.

Table 7.3: Dataset specific parameters settings

Dataset	X	y	$\oplus$	$\mathcal{V}$
Adult Census	Table C.2	Income level	$\geq 50K$	{World region●}
Bank Marketing	Table C.3	Subscribe	yes	{Semester★} {Quarter★}
Hotel Reservations	Table C.4	Booking status	cancelled	{Semester†} {Quarter†}
Flight Delays	Table C.5	Delay	yes	{From airport} {From US region◇}
University of Porto	Table C.6	Approval	fail	{Faculty} {Academic year, Scientific area}

Derived from: ● Native country, ★ Month; † { Arrival year, Arrival month }; ◇ From airport

7.2 Results of empirical evaluation

This section presents and discusses the empirical evaluation of consensus models, focusing on predictive performance, predictive performance loss, fidelity, interpretability, and completeness. Using the five datasets in Table 7.1, the analysis rigorously tests the core hypothesis. Specifically, it investigates whether the *Inmplode* framework can effectively construct consensus models that offer a concise yet representative summary of the distributed knowledge represented in the operational models. Critically, this data-driven evaluation is designed to reveal the generalizability of the results across diverse datasets or identify any dataset-specific characteristics.

7.2.1 Predictive performance

Given that consensus models are prediction models, an evaluation of their predictive capability is paramount. Consistent with the methodological formalization outlined in Section 4.8.1, the *Inmplode* framework employs an identical evaluation approach and uses the same data for the test sets for both operational and consensus models. This section details the predictive performance of consensus models, including the selected metrics, results, and discussion, while the next section focuses specifically on their comparison with operational models.

Table 7.4: Model evaluation parameter settings ($\mathcal{Z}$)

Set	Criterion	Target models	Metrics (Φ)
$\mathcal{Z}_1$	Performance	{Operational, Consensus}	{Accuracy, Precision, Recall, F1, κ , MCC}
$\mathcal{Z}_2$	Performance loss	{Consensus}	{Accuracy, Precision, Recall, F1, κ , MCC}
$\mathcal{Z}_3$	Fidelity	{Consensus}	{Accuracy, Precision, Recall, F1, κ , MCC}
$\mathcal{Z}_4$	Interpretability	{Operational, Consensus}	{Size, Complexity, Depth}
$\mathcal{Z}_5$	Completeness	{Consensus}	{FracCovered, GComp, DComp}

7.2.1.1 Metrics

Section 4.5.1 details the assessment of predictive performance metrics, including Accuracy, Precision, Recall, F1, Cohen’s Kappa (κ), and the Matthews Correlation Coefficient (MCC). While all metrics are considered, MCC (Equation 2.18) is prioritized as the primary comparative measure due to its robustness in class-imbalanced scenarios and its well-defined mathematical properties.

A key advantage of MCC lies in its fixed zero threshold, which distinguishes models that outperform random chance from those that do not. Unlike Accuracy, whose baseline performance varies depending on class distribution, making its interpretation more challenging, particularly in target variable class imbalanced datasets, MCC consistently assigns a value of zero to a model performing at chance level, regardless of class proportions. This strict grounded baseline fosters equal model comparisons across datasets with different class distributions.

Beyond its interpretability relative to Accuracy, MCC also differs from F1 in its treatment of both positive and negative class predictions. While F1 is defined as the harmonic mean of Precision and Recall, making it inherently focused on the positive class, MCC accounts for all four components of the confusion matrix (True Positives, True Negatives, False Positives, and False Negatives). This comprehensive formulation enables a more balanced evaluation of predictive performance, particularly in cases where misclassification of the negative class is also relevant.

Both MCC and κ are class-symmetric metrics bounded between -1 (total disagreement) and +1 (perfect agreement), with 0 indicating chance-level performance. Although κ and MCC share similarities in measuring agreement between predicted and actual labels, MCC is more stable in edge cases. For instance, when model evaluation yields zero true positives and zero true negatives, an outcome that can occur in highly imbalanced classification problems, κ becomes undefined, whereas MCC remains computationally valid (Chicco and Jurman, 2020; Chicco et al., 2021). This stability, combined with its resilience to class imbalance, makes MCC a reliable choice for rigorous performance evaluations, even in complex data contexts.

7.2.1.2 Results and discussion

To systematically evaluate the predictive performance of the three consensus model representations (Merged Table, Synthetic Data Trees, and Distributional Data Trees), Figure 7.1 presents their MCC scores across consensus models developed within the five datasets under investigation. The results reveal distinct performance patterns, with each representation exhibiting specific strengths and limitations depending on the dataset and experimental context.

In the Adult Census and Hotel Reservations datasets, the merged table representation consistently achieves higher predictive performance compared to both tree-based representations. In the Bank Marketing dataset, the merged table representation performs best for the first semester models, while in the second semester models, tree-based representations, particularly distributional data trees, show slightly stronger results. In the Flight Delays dataset, predictive performance varies across different geographical regions and consensus model representations. While merged table representations often achieve strong predictive performance, they also exhibit limitations in

coverage, failing to generate predictions for some test examples. In the University of Porto dataset, the synthetic data tree representation yields the highest predictive performance across most faculties, whereas merged table representations perform less effectively than tree-based models in several faculty-specific cases. Detailed tabular summaries of Accuracy, F1, and MCC results for each dataset and representation are provided in Appendix C.4.1.

Empirical results demonstrate that consensus models show significant variability in predictive performance across datasets and representations. In particular, converting merged decision tables into tree-based structures consistently reduces predictive accuracy. However, this performance gap should not be interpreted as a fundamental flaw in consensus modeling itself. Instead, it must be contextualized against the baseline performance of the original operational models used to derive the consensus models. Critically, the central uncertainty lies in whether this decline originates from information loss during structural conversion or reflects inherent limitations in the operational models. The following section addresses this through prediction loss analysis, which distinguishes between deficiencies introduced by consensus modeling methodology and pre-existing weaknesses in the original models.

7.2.2 Predictive performance loss

For a meaningful evaluation of the predictive performance of consensus models, it is essential to contextualize their evaluation in direct relation to the operational models they are designed to approximate and synthesize. It is important to emphasize that the primary objective of consensus models is not to replace existing operational models. Instead, the goal of this performance evaluation is to quantitatively assess the inherent performance-interpretability trade-off associated with adopting consensus models.

The trade-off in consensus modeling also parallels the well-established machine learning concepts of overfitting and underfitting, providing a useful framework for understanding its implications. Just as excessively complex models in traditional machine learning risk overfitting the training data, consensus models face a similar challenge, though in the opposite direction. While overfitting degrades generalization performance on unseen data, consensus models, intentionally designed with a focus on enhancing interpretability and knowledge consolidation, risk oversimplifying the intricate relationships captured by the original operational models. As a result, consensus models may exhibit underfitting characteristics, typically associated with overly simplistic model representations.

7.2.2.1 Metrics

An inherent aspect of developing consensus models, particularly those designed to enhance global interpretability and knowledge consolidation, is the potential for a trade-off with raw predictive performance when compared to the original, more specialized operational models. To systematically quantify this unavoidable trade-off, Section 4.8.2 introduces the concept of Performance Loss ($\Delta\phi$). Performance Loss is defined as the absolute difference in predictive performance between a

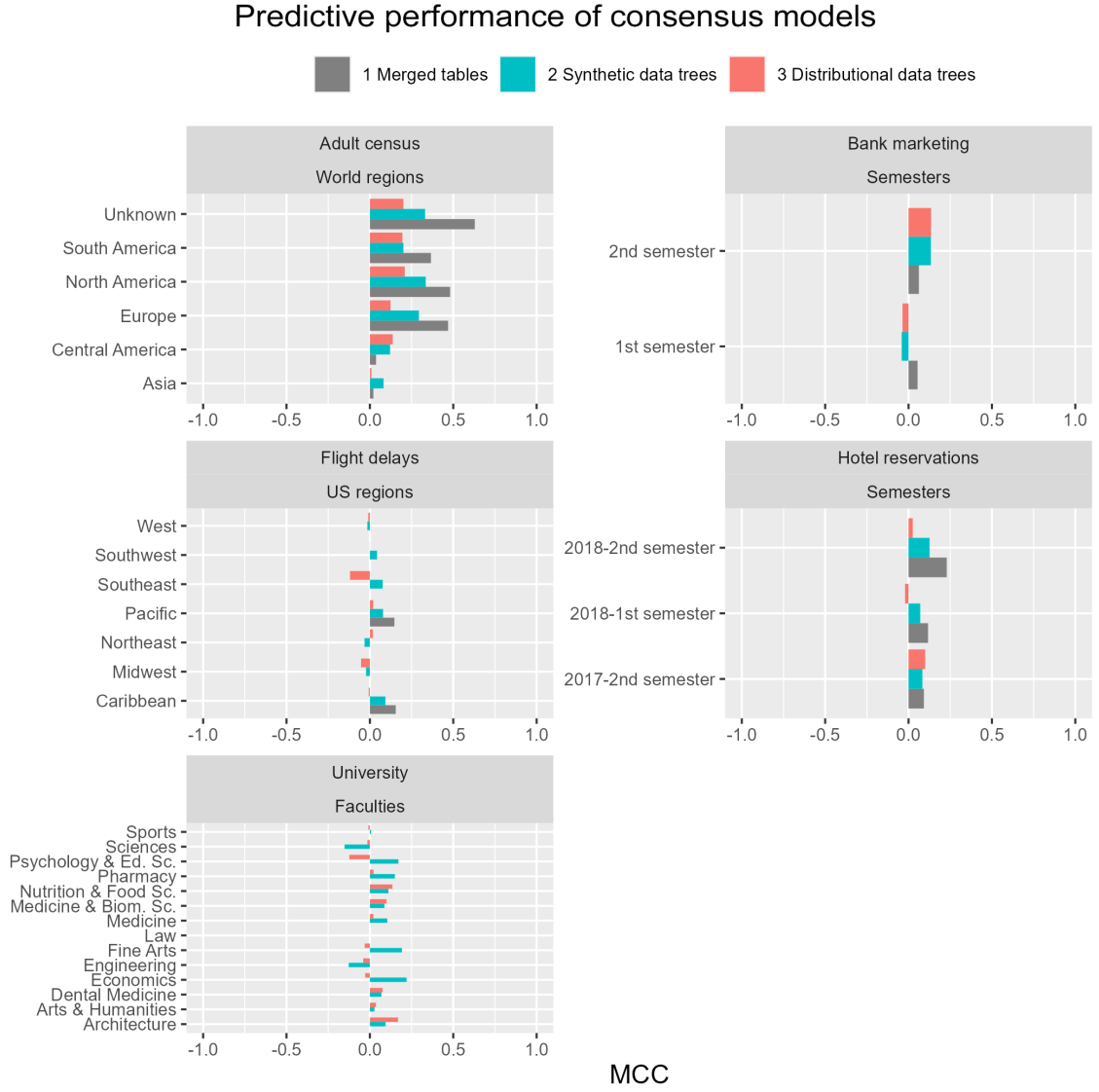


Figure 7.1: Results of the Predictive performance of consensus models measured with MCC

consensus model and an operational model (Equation 4.21). The Average Performance Loss ($\overline{\Delta\phi}$) of the consensus models is then computed by averaging this loss across all the operational models they represent (Equation 4.22).

While relative performance metrics, such as percentage change in performance relative to operational models, might appear intuitively appealing as measures of performance deviation, such relative metrics can be instable when the baseline operational model performance approaches zero. Therefore, the absolute difference in performance is adopted as a more immediate measure of performance deviation for this comparative analysis.

The computation of Average Performance Loss ($\overline{\Delta\phi}$) relies on identical test datasets for each paired comparison between a consensus model and its corresponding operational models, ensuring a fair and direct assessment of performance deviations. A value of $\overline{\Delta\phi} = 0$ signifies no observable

difference in predictive performance between the consensus model and the operational models. Conversely, $\overline{\Delta\phi} < 0$ indicates a decline in predictive performance for the consensus model relative to the models it represents. Although not the primary objective of developing consensus models, cases where $\overline{\Delta\phi} > 0$ suggest an improvement in predictive performance exhibited by the consensus model.

Consistent with the established rationale for prioritizing MCC in the overall predictive performance evaluation, as detailed in Section 7.2.1.1, $\overline{\Delta\text{MCC}}$ is also selected as the primary metric for quantifying predictive performance loss. However, to facilitate potential comparative analyses across different metric perspectives, the performance loss evaluation also includes $\overline{\Delta\text{Accuracy}}$ and $\overline{\Delta\text{F1}}$ results in Appendix C.4.2.

7.2.2.2 Results and discussion

To establish a necessary baseline for interpreting the performance loss of consensus models, the predictive performance of the underlying operational models is first examined (Figure 7.2). As expected, given the diversity of datasets and the complexities of the prediction tasks, operational model performance varies considerably. In particular, models trained on the Adult Census dataset achieve high MCC scores, indicating strong predictive capabilities for this task. In contrast, models trained on the Flight Delays and University of Porto datasets exhibit considerably weaker performance, suggesting that these datasets pose greater challenges for the selected modeling approach. The Bank Marketing and Hotel Reservations datasets fall into an intermediate performance range. This heterogeneity in baseline operational model performance is more than a descriptive observation; it serves as a critical contextual element for interpreting subsequent analyses of consensus model performance losses. Specifically, determining whether observed losses indicate a degradation from a strong baseline or merely reflect the propagation, or even amplification, of pre-existing limitations in the operational models is essential, particularly when applied to inherently difficult prediction tasks.

As illustrated in Figure 7.3, consensus models generally exhibit a decline in performance ($\overline{\Delta\text{MCC}} < 0$) compared to the operational models they represent. However, the extent of performance loss varies across representations and datasets. As expected, tree-based representations (synthetic and distributional trees) tend to experience greater performance declines than merged tables. This effect may stem from the inductive bias of tree-based models, which favor generalization over retaining the more granular decision rules present in merged tables. Consequently, this generalization can lead to model oversimplification, sacrificing detailed predictive information preserved in the tabular structure of merged tables.

Merged table representations, by directly aggregating and preserving the original decision logic, tend to exhibit smaller performance declines, indicating a higher degree of knowledge representation of the operational models. However, their applicability is limited by the coverage limitations of the dataset. In the Flight Delays and University of Porto datasets, the near-zero test set coverage of merged tables makes performance loss comparisons unreliable. This highlights the need to assess both predictive performance and the completeness of the decision space.

The substantial performance losses observed for tree-based consensus models on the University of Porto dataset may not solely reflect representational limitations but could also stem from the restricted and potentially biased subset of the test set covered by the corresponding merged tables. Therefore, performance loss interpretations in coverage-limited cases should be approached with caution.

7.2.3 Fidelity

Fidelity evaluation assesses the extent to which a consensus model can mimic the predictions of the operational models. In essence, it evaluates whether the consensus model accurately represents the same knowledge as the underlying operational models. Although fidelity, by definition, does not directly assess predictive accuracy against ground truth, it is expected to exhibit a strong positive correlation with predictive performance. Specifically, if a consensus model closely replicates the predictions of an operational model with high predictive accuracy, it is reasonable to assume that the consensus model will also achieve a comparable level of performance. Nevertheless, fidelity acts as a proxy for gauging how well the consensus representation preserves the original operational models' decision-making logic.

7.2.3.1 Metrics

Formally, fidelity quantifies the alignment between the predictive outputs of two models, measuring the extent to which one replicates or approximates the other's predictions. Unlike predictive performance metrics, which evaluate correctness against ground truth labels, fidelity focuses exclusively on inter-model consistency, independent of accuracy relative to ground truth. Fidelity can be assessed symmetrically, measuring mutual agreement between models, or asymmetrically, comparing a designated proxy model against a predefined reference. In the evaluation of consensus models, an asymmetric approach is adopted, where each consensus model is systematically compared against its corresponding operational models, which serve as the reference for knowledge representation.

To quantify the agreement between consensus and operational model predictions, standard predictive performance metrics are adapted by prefixing them with *Fid* to denote their application in fidelity assessment: *FidAccuracy*, *FidF1*, and *FidMCC*. *FidAccuracy* measures the percentage of identical predictions between the consensus and operational models. *FidF1* evaluates agreement focusing on positive class predictions. *FidMCC*, consistent with its role in predictive performance evaluation, assesses balanced agreement across all four confusion matrix categories, ensuring robustness even under class imbalance. A *FidMCC* score of 1 signifies perfect alignment with the operational model.

In line with the methodological framework established for evaluating both predictive performance and predictive performance loss, the MCC is selected as the primary metric for measuring fidelity, referred to as *FidMCC* (as detailed in Section 7.2.1.1). A more comprehensive overview of the results, including *FidAccuracy*, *FidF1*, and *FidMCC*, are presented in Appendix C.4.3.

Empirical Evaluation of *Inmplode*

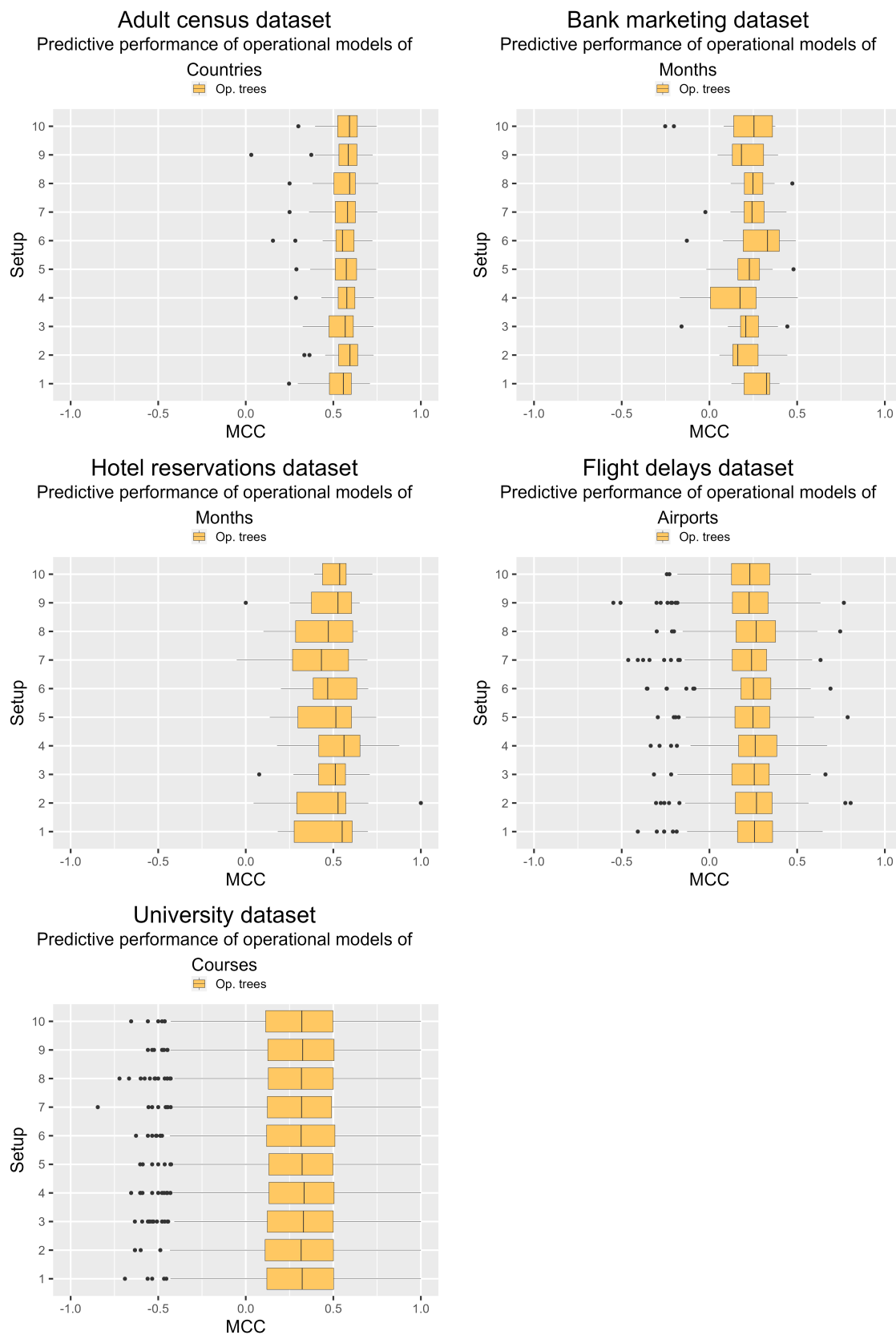


Figure 7.2: Results of the Predictive performance of operational models measured with MCC

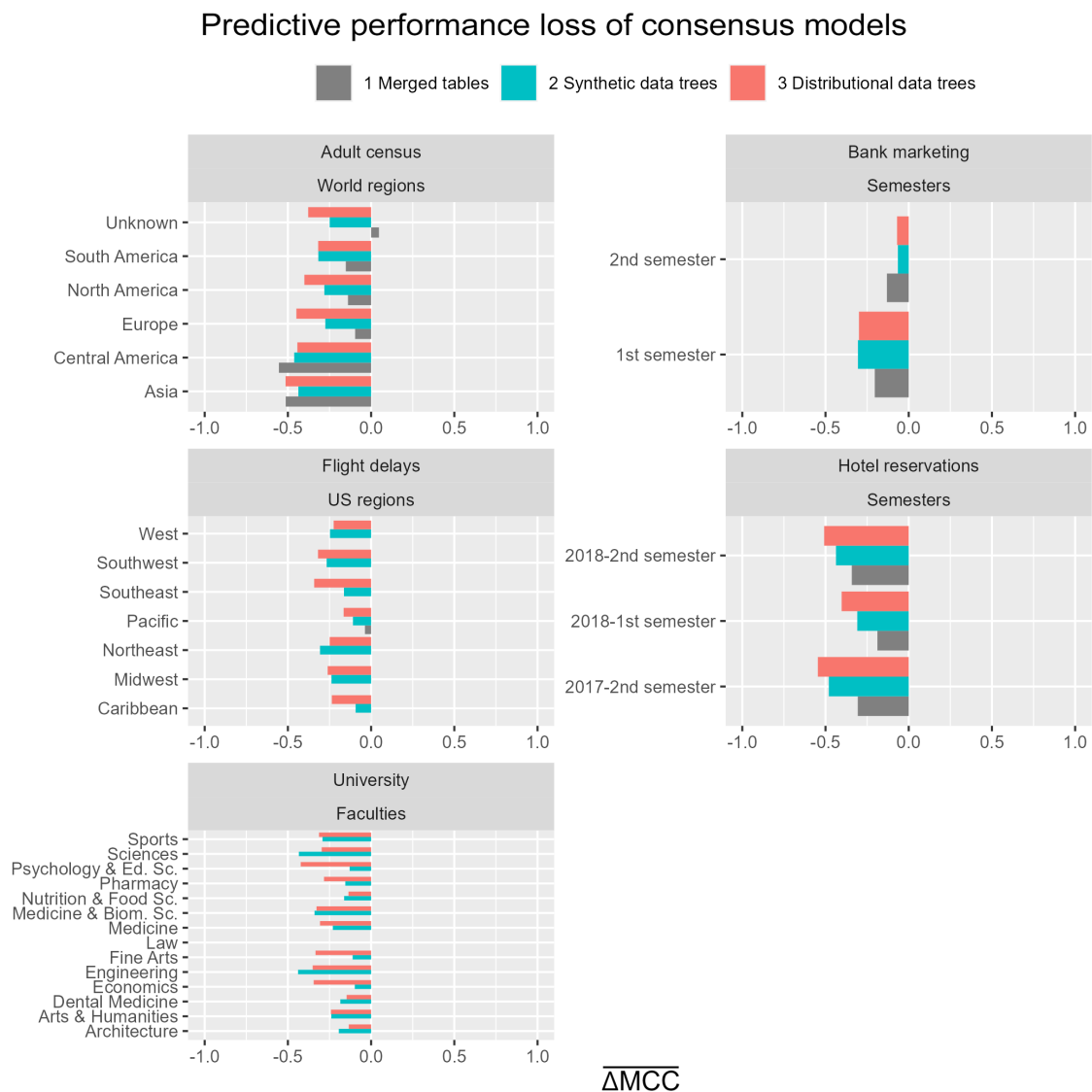


Figure 7.3: Results of the Predictive performance loss of consensus models measured with $\overline{\Delta\text{MCC}}$

7.2.3.2 Results and discussion

Figure 7.4 illustrates the variability of consensus model fidelity, quantified by FidMCC , across the consensus models developed in each dataset and model representations evaluated. Merged table representations exhibit the highest fidelity scores across most datasets. In particular, in the Adult Census dataset, a few merged table consensus models achieve fidelity scores that indicate a close alignment with the predictive outputs of their corresponding operational models. Conversely, tree-based representations, including both synthetic and distributional data trees, consistently achieve comparatively lower fidelity scores across the datasets. This pattern suggests a less direct replication of the operational models' decision logic by tree-based consensus representations compared to merged tables. When comparing the tree-based representations, synthetic data trees generally exhibit slightly higher fidelity scores than distributional data trees. A similar trend in fidelity

scores across model representations is observed in the Bank Marketing dataset, further illustrating the general patterns in fidelity performance across the evaluated datasets.

Fidelity evaluation in the Flight Delays and University of Porto datasets is significantly affected by test set coverage limitations, particularly for merged tables. This limitation makes fidelity comparisons for merged tables in these datasets unreliable. As a result, the analysis in these cases focuses primarily on tree-based representations, where synthetic data trees again demonstrate higher fidelity than distributional data trees. Negative FidMCC values in a few consensus models indicate not just low agreement but systematic disagreement, meaning the consensus model predictions are inversely correlated with those of the operational models, which can defeat their purpose.

The results of fidelity align with patterns observed in predictive performance evaluation, reinforcing the trade-off between accuracy and interpretability in consensus models. While tree-based representations exhibit lower fidelity than merged tables, this difference reflects the inherent cost of improving interpretability through tree representations. These findings emphasize that consensus models are not intended to replace high-performing operational models but rather to provide interpretable approximations that reveal the core decision logic within complex models. This trade-off is evident both in the comparison between tables and trees and within tree-based representations themselves. Synthetic data trees, despite only slightly higher fidelity than distributional trees, highlight differences in knowledge representation, while merged tables, although achieving higher fidelity in some cases, do so at the expense of the interpretability that tree-based models aim to improve.

7.2.4 Interpretability

Model interpretability refers to the ease with which a human can understand a model’s decision-making processes and overall behavior. Unlike predictive performance, which benefits from well-established and widely accepted evaluation metrics, interpretability lacks standardized quantitative measures. As a result, direct human evaluation, while valuable, is resource-intensive and difficult to scale for systematic model comparison.

To address this challenge, objective complexity metrics are employed as proxies for interpretability. These metrics enable an automated, quantitative analysis of structural characteristics that are expected to correlate with human understanding, reducing the reliance on time-consuming, subjective assessments. This approach facilitates a systematic comparison of different model representations, recognizing that while complexity metrics provide only an indirect measure, they offer a practical and scalable method for assessing this inherently human-centered quality.

7.2.4.1 Metrics

To evaluate model interpretability, three complementary complexity metrics are employed: Size, Dimensionality, and Depth. Size (Equation 3.16) is defined as the total count of decision rules within the model. This metric functions as a direct indicator of decision-making complexity, on

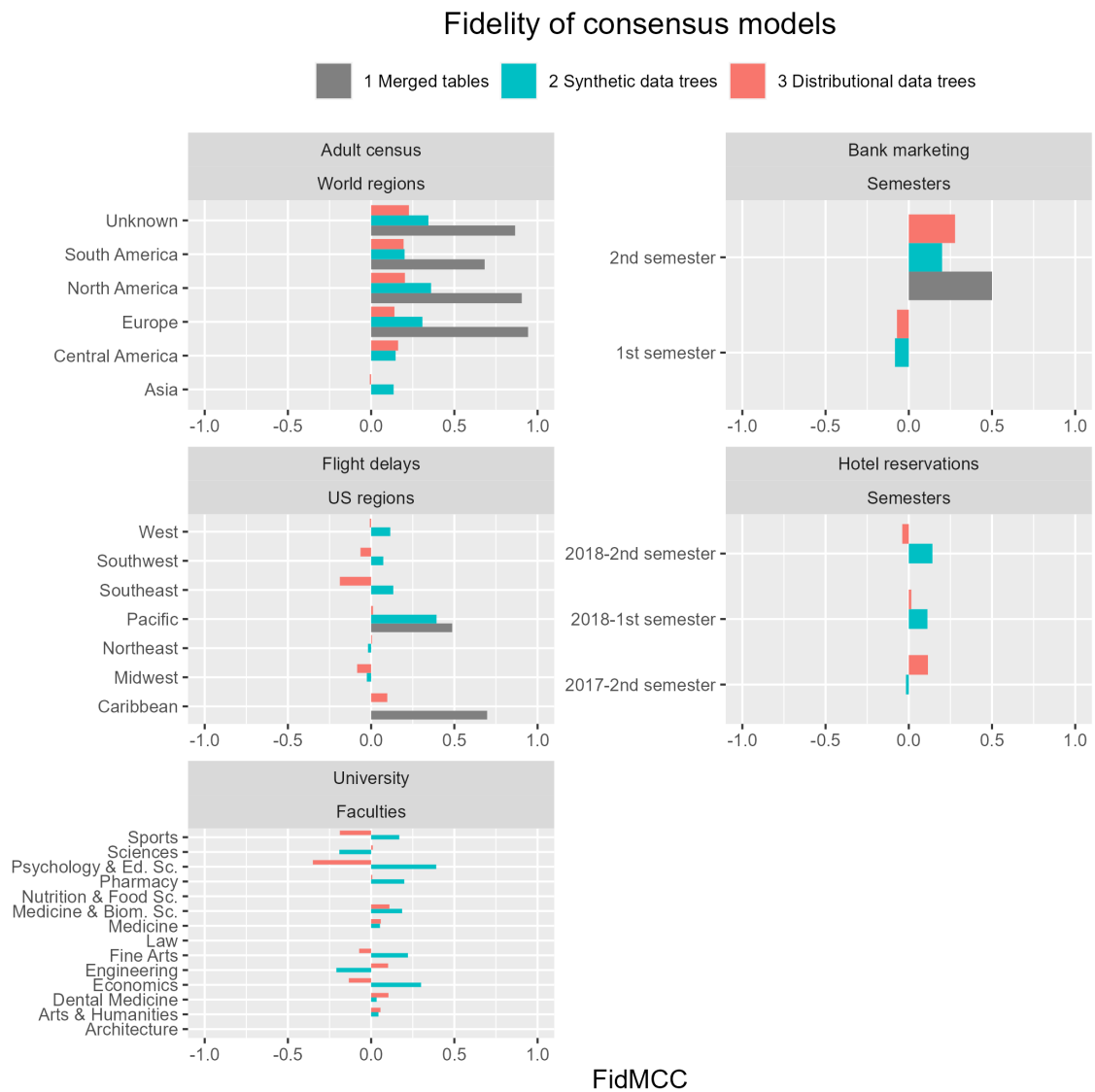


Figure 7.4: Results of the Fidelity of consensus models measured with FidMCC

the principle that larger models, characterized by more extensive rule sets, tend to be inherently more challenging for human comprehension and interpretation. Dimensionality (Equation 3.20) quantifies the number of distinct independent variables used within the model's decision rules. This metric ranges from zero, representing a single-rule model with no conditions, up to the total count of independent variables available in the training dataset. It is generally accepted that models employing a smaller subset of input variables are intrinsically easier to interpret. For consensus models represented as decision trees, Depth (Equation 3.19) is additionally considered, defined as the number of hierarchical levels within the tree. Increased tree depth implies more intricate and nested decision paths, which, in turn, introduce greater cognitive complexity and consequently reduce overall model interpretability.

7.2.4.2 Results and discussion

To provide essential context for evaluating the interpretability of consensus models, a systematic characterization of the intrinsic complexity of the original operational models themselves is presented, as in predictive performance evaluation. Figures 7.5, 7.6, and 7.7 present summary distributions of Size, Dimensionality, and Depth, respectively, across the ten training setups per dataset. The interpretability of the operational models acts as the baseline, enabling a direct assessment of the degree of simplification achieved with the consensus model.

Results of Size reveals substantial inter-dataset variability in the number of decision rules, while intra-dataset variation across training setups remains comparatively minimal. In particular, operational models trained on the Adult Census dataset exhibit the most extensive rule sets, ranging from dozens to, in some instances, thousands of rules, indicative of high complexity. In contrast, models for Bank Marketing and Flight Delays datasets rarely exceed 20 rules, with a few cases reaching hundreds. Hotel Reservations models typically exhibit moderately sized rule sets, averaging around 30 rules, while University of Porto models are predominantly characterized by very compact rule sets, generally below 10 rules, albeit with outliers reaching into the dozens. This contrast suggests that the inherent complexity required to model the Adult Census dataset significantly exceeds that of the Bank Marketing, Flight Delays, and University of Porto datasets.

Dimensionality show stability across training setups within each dataset, but significant variation across datasets. Operational models for the Adult Census dataset consistently use a substantial proportion of the available independent variables, ranging from 8 to 9 out of 12. Bank Marketing models, in contrast, rely on a smaller subset, typically 2 to 5 variables out of 15, and Hotel Reservations models use a range of 5 to 10 out of 18. Flight Delays models are characterized by very low dimensionality, never exceeding 4 out of 5 variables, and University of Porto models exhibit similarly sparse variable use, employing only 1 to 5 out of 26 available variables. This pattern underscores the varying degrees of variable use across datasets, with Adult Census requiring a broader set of variables, while Flight Delays and University of Porto models effectively operate with a highly selective subset of independent variables.

Depth, in contrast to Dimensionality, exhibits greater variability both across datasets and, to a lesser extent, between training setups within datasets. Operational models for the Adult Census dataset display the greatest depth, averaging around 10 levels but occasionally extending to several dozen, indicative of more complex trees. Bank Marketing trees, conversely, rarely exceed a depth of five levels. Hotel Reservations trees span a range of 10 to 20 levels, while Flight Delays and University of Porto trees consistently demonstrate shallower depths, typically within the 1 to 7 and 1 to 6 level ranges, respectively, and exhibiting greater stability across training setups. Despite this variability, a consistent and expected trend emerges: simpler operational models, characterized by reduced Size and Dimensionality, tend to also exhibit shallower Tree Depth, particularly evident in the Bank Marketing and University of Porto datasets, suggesting a congruence across these complexity metrics.

Synthesizing these complexity metrics reveals pronounced structural heterogeneity in the inherent interpretability of operational models across the datasets. For example, the Hotel Reservations dataset consistently presents the highest level of complexity across all metrics, characterized by expansive rule sets, multi-variable feature interactions, and deeper decision trees. In stark opposition, the Flight Delays dataset consistently demonstrates minimal complexity, characterized by sparse decision logic, low dimensionality, and shallower trees. This divergence in inherent model complexity across datasets underscores the critical point that model interpretability is not an absolute property, but rather is fundamentally domain-specific, contingent on the inherent complexity of the underlying data and the nature of the predictive task itself.

Figure 7.8 summarizes the Size of consensus models as rows in merged tables and leaves in decision trees. A consistent and pronounced trend is evident: table representations, across datasets, consistently exhibit the largest rule sets, indicating the highest structural complexity. Conversely, the conversion to tree representations systematically yields a marked reduction in Size, with only minor exceptions observed. This structural simplification is most observed in the University of Porto dataset, where tree-based consensus models achieve a reduction in rule counts by orders of magnitude, representing a substantial decrease, often by factors of ten or more, in model complexity. Furthermore, within the tree-based representations, distributional data trees consistently produce the leanest tree structures, characterized by a lower number of nodes and, consequently, enhanced interpretability relative to synthetic data trees. This suggests that the distributional data tree approach is particularly effective in distilling a simplified yet informative representation of the operational models' decision logic.

Figure 7.9 illustrates the Dimensionality of consensus models. Mirroring the trends observed for Size, merged table representations consistently exhibit the highest Dimensionality, while tree-based representations demonstrate a significant reduction in the number of variables used in the models. This disparity is again most pronounced in the University of Porto dataset, where tree-based models reduce the number of variables by approximately five, on average, compared to merged table counterparts. Analogous to trends observed in Size, distributional data trees persistently result in consensus models with fewer variables than synthetic data trees, reinforcing the pattern of distributional trees offering greater simplification. However, a divergence emerges in contrast to the reduction observed in Size: consensus models, regardless of representation (tables or trees), generally retain the dimensionality characteristics of the original operational models. This suggests that while consensus models achieve a significant simplification in the granularity of decision rules (Size), they largely preserve the variable selection decisions inherent in the operational models, suggesting that the core set of relevant independent variables is maintained, even as the decision logic is simplified.

Figure 7.10 depicts the Depth of tree-based consensus models. Aligning with the consistent trends observed for both Size and Dimensionality, distributional data trees systematically exhibit shallower tree structures compared to their synthetic data tree counterparts. While all consensus tree models maintain a moderate range of depth across datasets, a clear and consistent pattern is observed: irrespective of representation (synthetic or distributional), consensus trees achieve

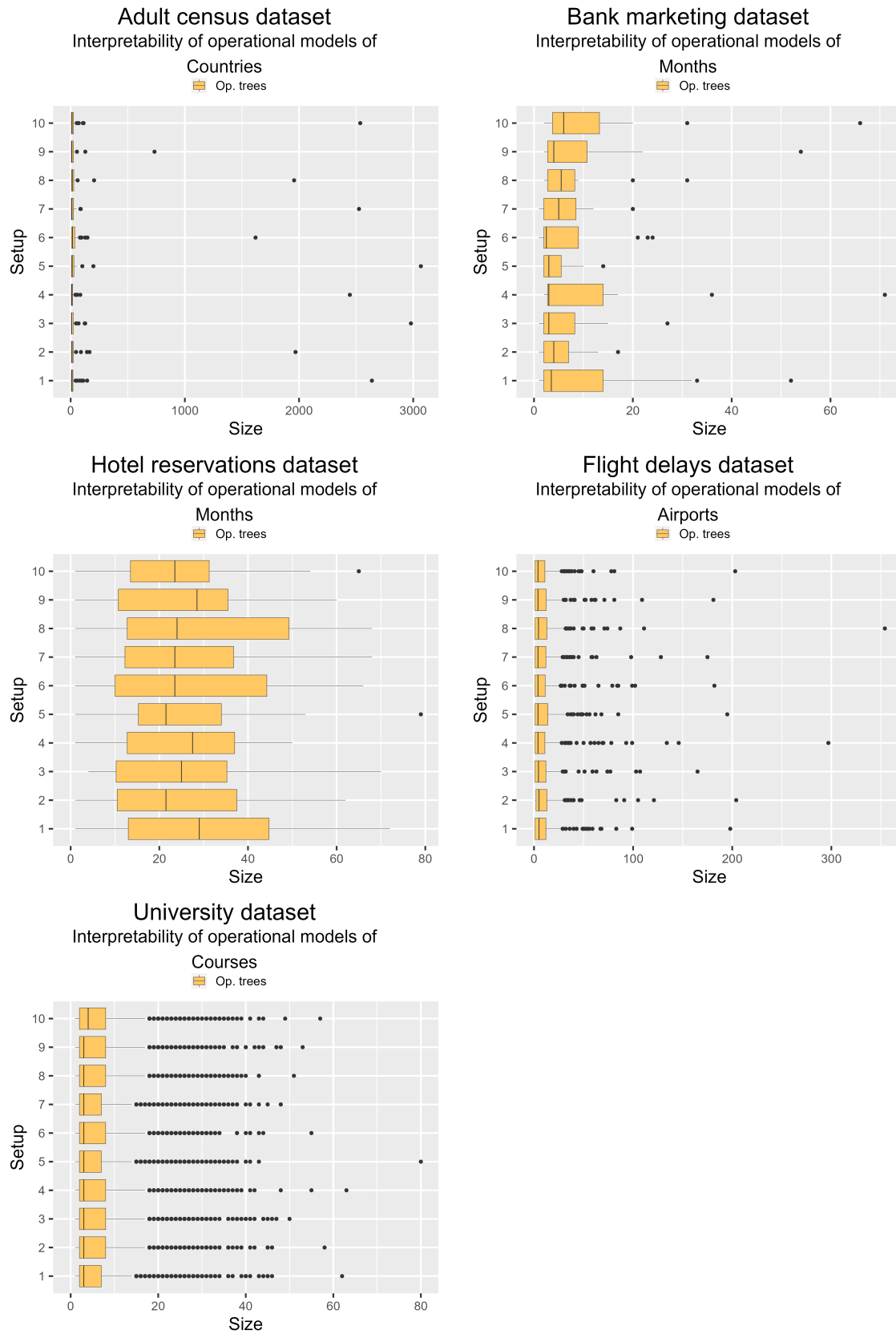


Figure 7.5: Results of the Interpretability of operational models measured with Size

Empirical Evaluation of *Inmplode*

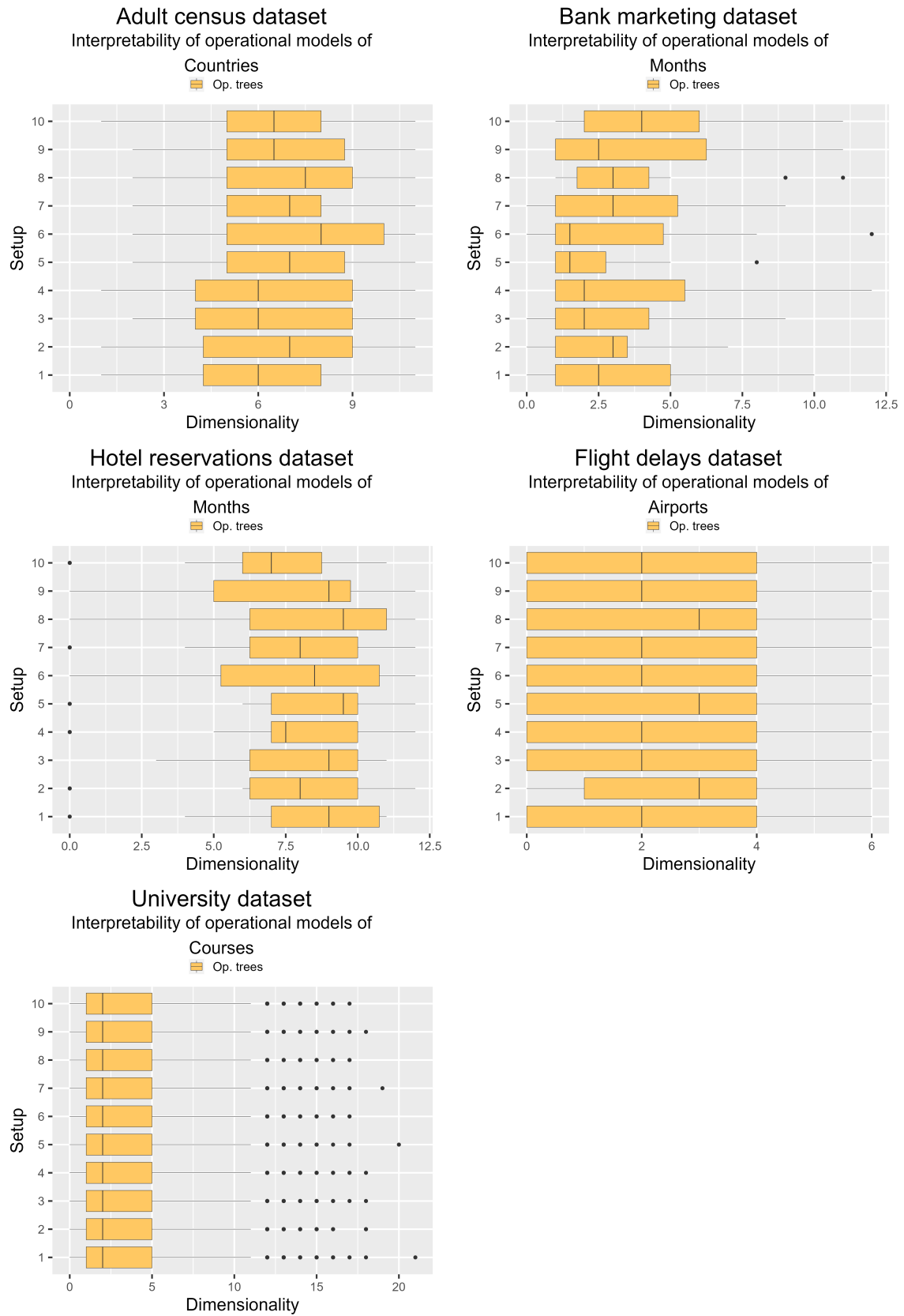


Figure 7.6: Results of the Interpretability of operational models measured with Dimensionality

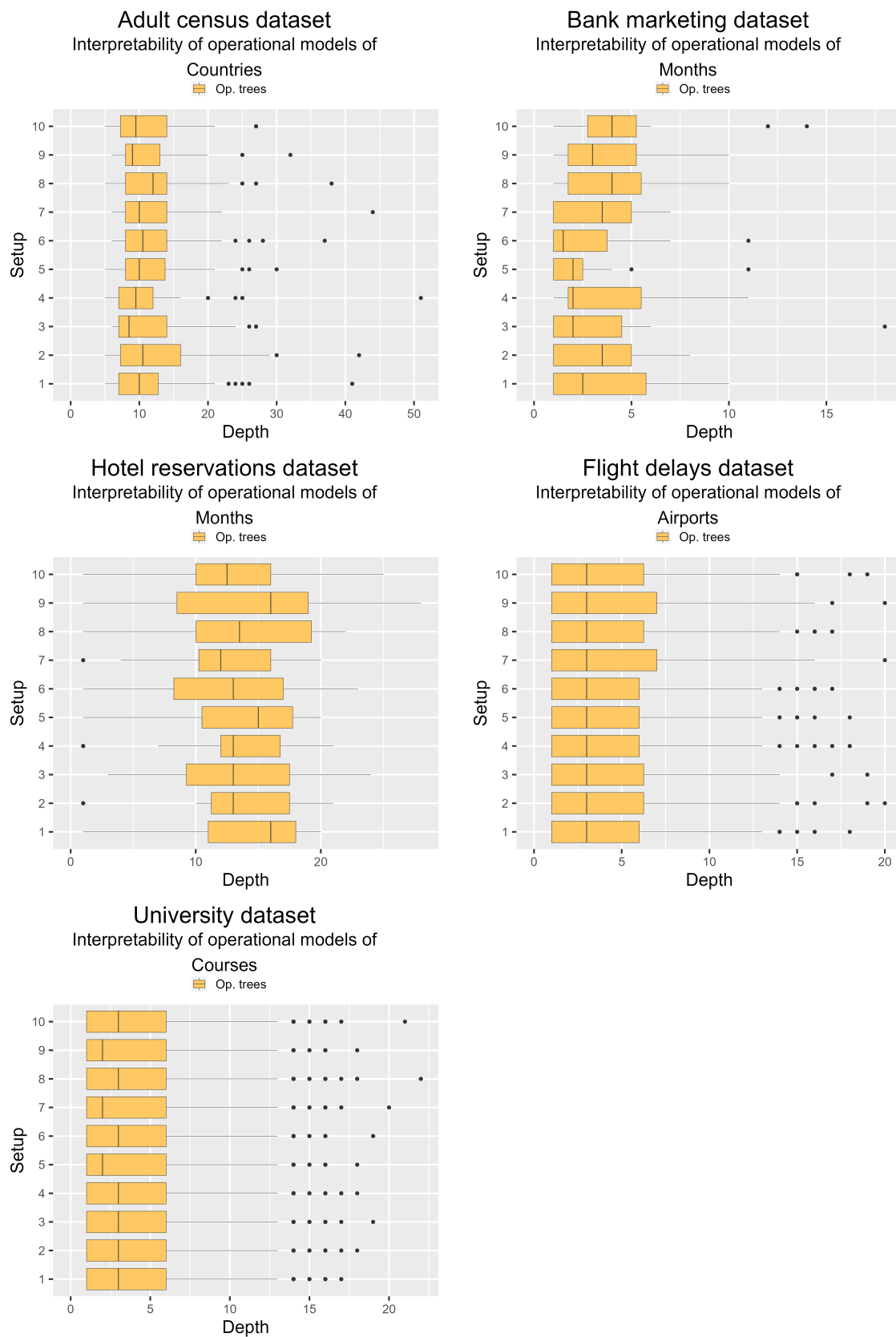


Figure 7.7: Results of the Interpretability of operational models measured with Depth

a substantial and consistent reduction in Depth relative to the original operational models. This further underscores the effectiveness of consensus models in achieving structural simplification, not only in terms of rule quantity but also in terms of the hierarchical complexity of the decision-making process, as captured by tree depth. Detailed interpretability results for consensus models are provided in Section C.4.4 for further examination.

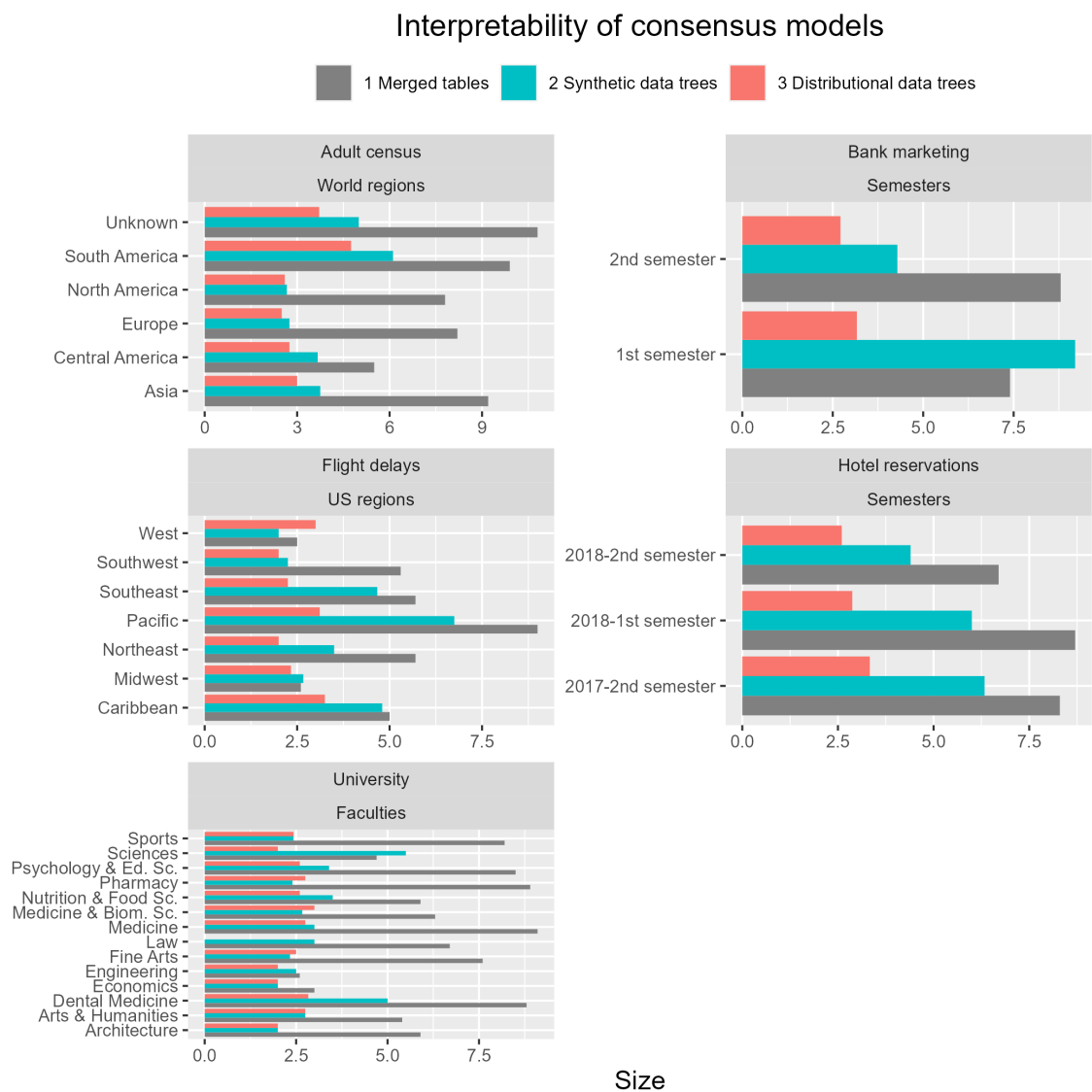


Figure 7.8: Results of the Interpretability of consensus models measured with Size

Synthesizing the results across all three complexity metrics, Size, Dimensionality, and Depth, reveals systematic patterns in the interpretability of consensus models. Consensus models represented as merged tables consistently exhibit the highest structural complexity across all datasets, as consistently reflected in their more expansive rule sets and higher number of variables. In direct contrast, tree-based representations, and particularly distributional data trees, demonstrably exhibit markedly lower complexity across all metrics, effectively rendering them as significantly

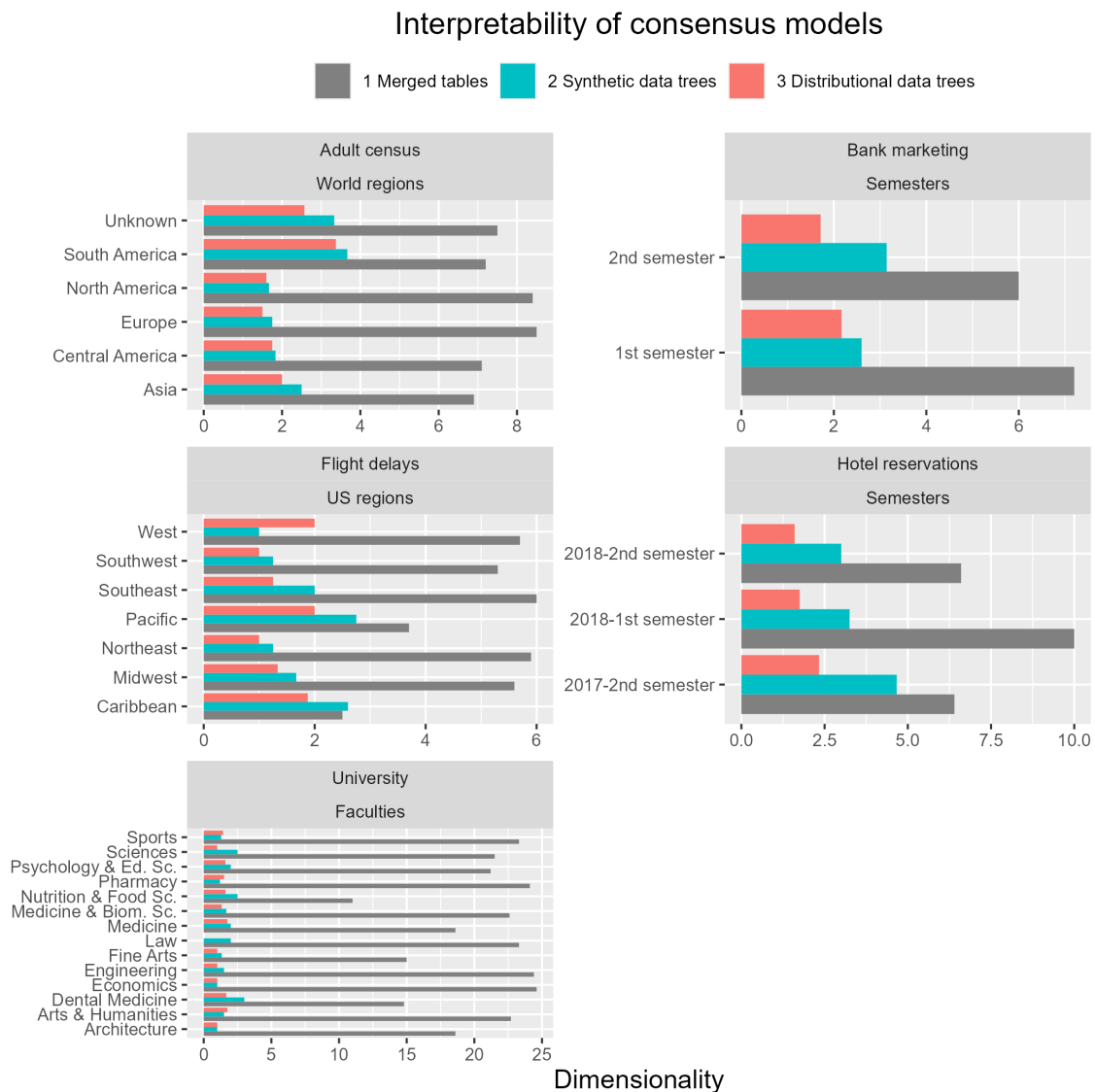


Figure 7.9: Results of the Interpretability of consensus models measured with Dimensionality

more interpretable representations of the operational models' decision logic. Moreover, distributional data trees consistently demonstrate systematically reduced Size and shallower Depth when compared to synthetic data tree variants, highlighting their higher simplification capability within the tree-based family. Furthermore, the pronounced and consistent reduction in both rule counts and tree depth in consensus models, when contrasted with the original operational models, aligns with the inherent trade-off between model complexity and interpretability, central to the purpose of consensus models.

7.2.5 Completeness

Model completeness refers to the extent to which a model covers the decision space defined by its independent variables. Instead of being viewed as a binary characteristic (complete or incomplete),

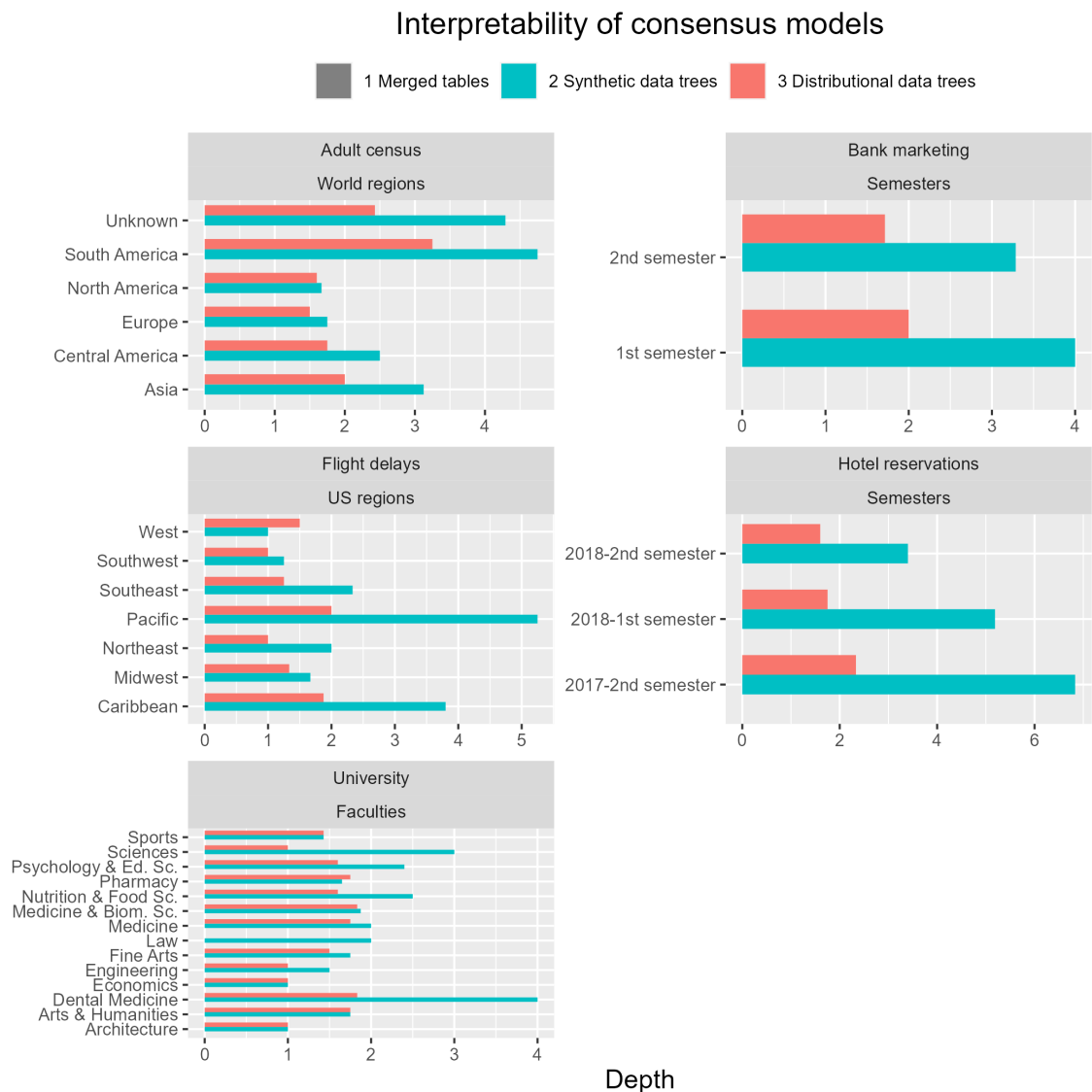


Figure 7.10: Results of the Interpretability of consensus models measured with Depth

model completeness reflects varying degrees of coverage. At the upper bound, maximum completeness refers to a model that is capable of generating predictions for every possible combination of values across its independent variables domain, ensuring exhaustive coverage of the decision space. Conversely, partial completeness arises when a model's predictive capacity is restricted to only a subset of these combinations, inherently resulting in coverage gaps within the decision space. At the theoretical lower bound, the concept of zero completeness describes a model entirely lacking in decision logic, a purely hypothetical construct with no practical empirical realization.

Operational models, decision trees in this context, inherently achieve maximum completeness due to their recursive partitioning approach. This method systematically divides the decision space into mutually exclusive and collectively exhaustive regions, ensuring comprehensive coverage across the entire domain of independent variables. In contrast, consensus models tend to

experience a decline in completeness as a result of the structural simplification processes involved in their generation, particularly during the merging process. This reduction in completeness highlights a critical trade-off: while consensus models enhance interpretability and may improve generalization, their inherent incompleteness can lead to underspecification in real-world applications, especially in scenarios where unrepresented yet valid combinations of values of independent variables may arise.

7.2.5.1 Metrics

Model completeness is quantitatively assessed using three distinct metrics, each ranging within the interval $[0; 1]$, where one signifies maximum completeness. Fraction Covered (FracCovered, Equation 3.21) measures the proportion of instances in a given test set for which a model can generate a prediction. While it provides a direct assessment of empirical coverage, FracCovered is inherently limited by its dependence on the availability of a test set. This reliance restricts its generalizability and applicability beyond the specific evaluation scenario defined by the test set.

To address this limitation and enable a more robust assessment of completeness, two novel metrics are introduced: Geometrical Completeness (GComp) and Distributional Completeness (DComp). GComp (Equation B.7) evaluates completeness by explicitly identifying and quantifying coverage gaps within the decision space defined by the independent variables domains. In contrast, DComp (Equation B.9) prioritizes coverage of regions with higher data density to align model coverage with the intrinsic distributional properties of the data. A key advantage of both GComp and DComp is their independence from a specific test set, addressing the limitations of FracCovered and providing a more generalizable perspective on model completeness. As a supplementary research contribution to this thesis, the methodological foundations, including mathematical formulations and algorithmic implementation of both metrics are detailed in Appendix B.

7.2.5.2 Results and discussion

Figure 7.11 presents the results for FracCovered, revealing pronounced variations in empirical completeness across the three consensus model representations. As a general trend, merged table representations tend to exhibit lower FracCovered values compared to both tree-based representations. In particular, the Flight Delays dataset displays particularly high variability in FracCovered for merged tables, ranging from models achieving relatively high coverage to others approaching or even reaching zero FracCovered. A similar, though less extreme, trend is observed in the University of Porto dataset, where the majority of merged tables exhibit very low FracCovered, effectively limiting the reliability of predictive performance evaluations for these representations in these datasets, as previously discussed. Among tree-based representations, distributional data trees consistently achieve the highest FracCovered values across datasets, indicating broader test set coverage, while synthetic data trees demonstrate slightly, but consistently, lower FracCovered values. However, it is noteworthy that in the University of Porto dataset, both tree representations

converge to exhibit similar, and relatively lower, *FracCovered* values compared to their performance in other datasets, possibly suggesting dataset-specific factors limiting empirical coverage even for tree-based models.

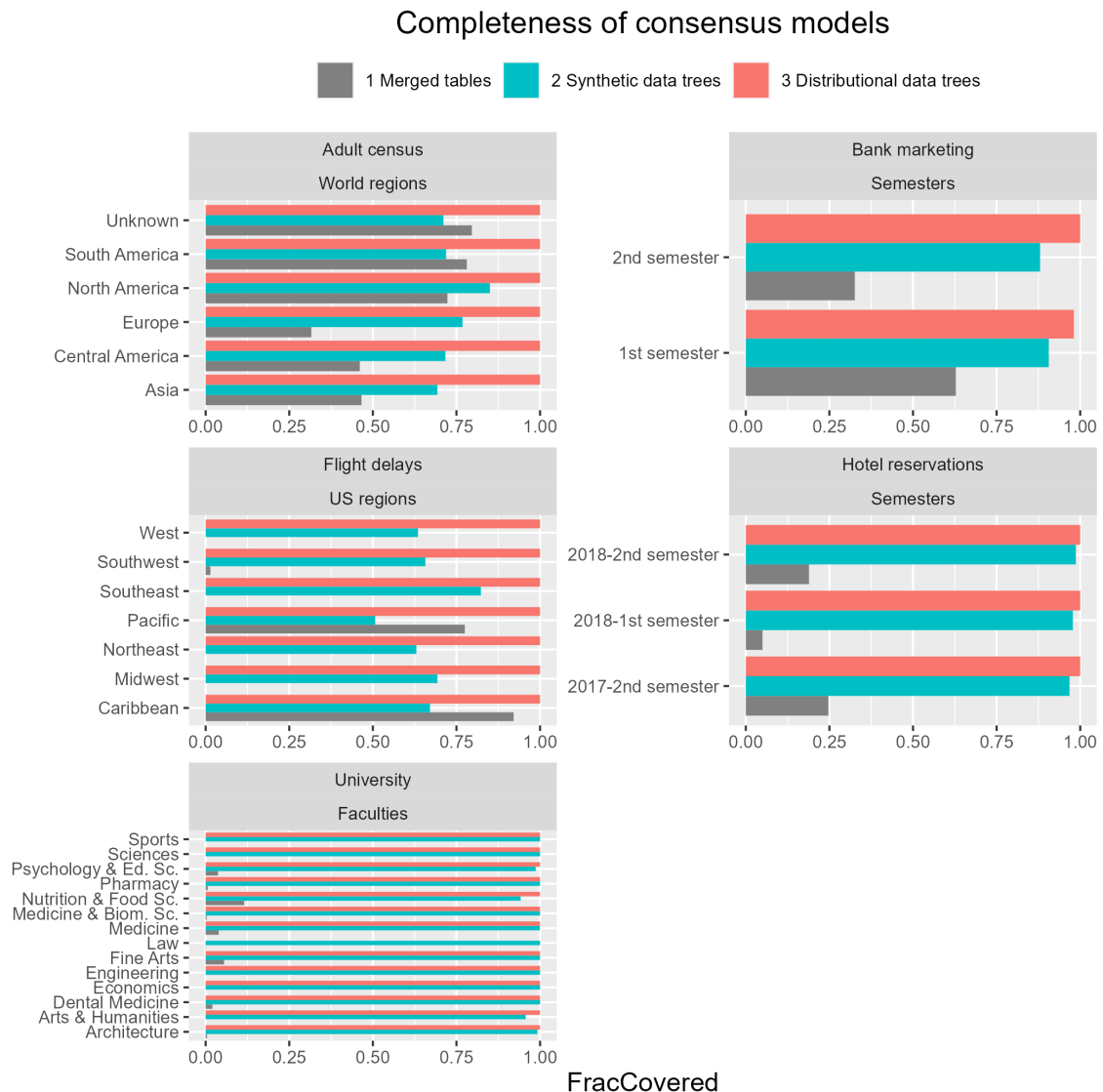


Figure 7.11: Results of the Completeness of consensus models measured with Fraction covered

Figure 7.12 illustrates the results for Geometrical Completeness (GComp). Mirroring the results for *FracCovered*, a consistent pattern emerges across the majority of datasets: merged table representations generally exhibit the lowest GComp, indicating broader coverage gaps within the uniformly weighted decision space. While in the Adult Census dataset, merged tables surpass synthetic data trees in GComp for a limited subset of consensus models, this appears to be an exception rather than the rule. In the Bank Marketing dataset, the GComp of merged tables stabilizes at a relatively low value of approximately 0.25, suggesting substantial and consistent coverage limitations. For the Flight Delays dataset, merged tables exhibit highly variable GComp, ranging from

performance comparable to Adult Census models in some instances to near-zero completeness in others, highlighting inconsistency. In the Hotel Reservations and University of Porto datasets, the GComp of merged tables consistently approaches or remains at zero, indicating a near-complete absence of coverage in these more complex and potentially higher-dimensional decision spaces.

In direct contrast to merged tables, tree-based representations, and particularly distributional data trees, demonstrate significantly higher GComp across all datasets. A clear trend emerges when comparing tree-based representations: distributional data trees consistently achieve the highest GComp values, indicating higher and more uniform coverage of the decision space. In every dataset, distributional data trees demonstrate greater coverage compared to both synthetic data trees and merged table representations. While synthetic data trees maintain GComp values generally above 0.5 across datasets, indicating moderate coverage, they consistently fall short of the maximum completeness levels frequently achieved by distributional data trees, suggesting a less complete and potentially more geometrically biased coverage pattern.

Figure 7.13 presents the results for Distributional Completeness (DComp). The patterns observed for GComp are largely mirrored and even amplified in DComp. Distributional data trees, which consistently achieve maximum values (1.0) in GComp, similarly reach maximum values in DComp. This confirms their exceptional ability to fully cover both the decision space and the underlying data distribution. Furthermore, and as anticipated given DComp’s data-density weighting, the DComp values for both merged tables and synthetic data trees are generally numerically higher compared to their corresponding GComp values. This consistent numerical increase stems from the inherent differences in metric formulation: distributional completeness specifically emphasizes coverage in regions with higher data density, thereby favoring models that focus on these data-rich areas, even if their overall uniform geometrical coverage is comparatively lower.

Synthetic data trees, in turn, consistently reach higher DComp values across datasets compared to merged tables, mirroring their relative ranking in GComp. However, it is noteworthy that the performance gap between synthetic and distributional data trees, while still present, is comparatively less pronounced in DComp than in GComp. This reduction in tree-based representations in DComp suggests that while distributional data trees excel in uniform geometrical coverage, synthetic data trees also provide effective coverage when weighted by data density. This is especially evident when compared to the limited coverage of merged tables. The relationship between tree-based representations in DComp follows a similar relative trend to that observed in GComp, but the magnitude of the performance losses associated with synthetic data trees, compared to distributional data trees, is somewhat attenuated when evaluated using DComp. Detailed tabular summaries of completeness metrics results for each dataset and representation are provided in Appendix C.4.5.

Despite their distinct computational methodologies, the three completeness metrics, FracCovered, GComp, and DComp, converge on two conclusions. Firstly, across all metrics and datasets, tree-based representations, consistently produce more complete consensus models compared to merged table representations. Furthermore, within the tree-based representations, distributional data trees consistently have the highest levels of completeness, frequently reaching maximum

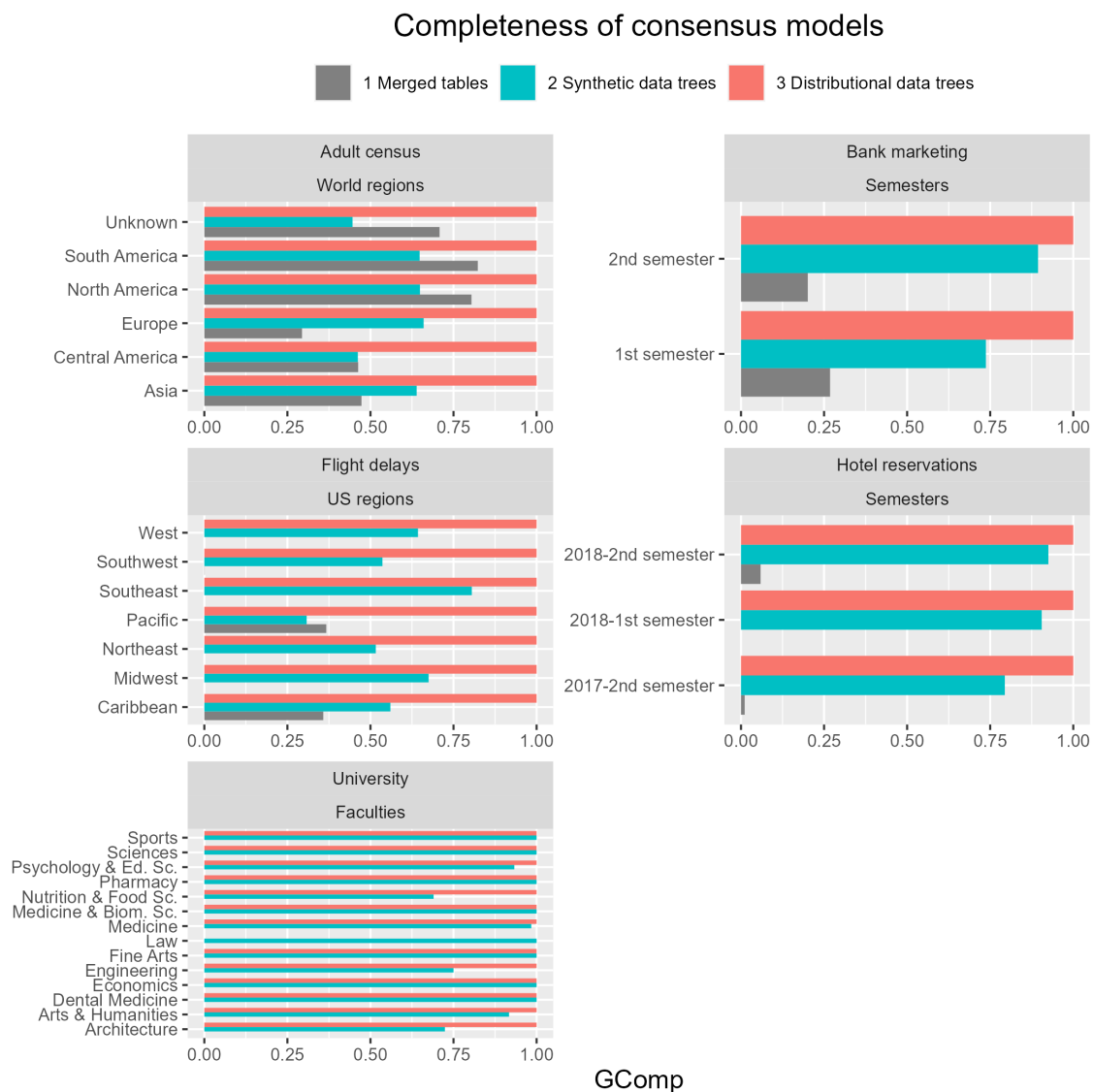


Figure 7.12: Results of the Completeness of consensus models measured with Geometrical completeness

completeness values across metrics. This demonstrates their inherent ability to effectively generalize decision rules across the entire decision space by explicitly incorporating information of the underlying data density distribution within that space.

Secondly, Geometrical Completeness, which explicitly evaluates coverage uniformly across the entire decision space, consistently exhibits quantitatively lower values for merged table and synthetic data tree representations compared to distributional data trees. However, even GComp consistently indicates maximum completeness (values approaching 1.0) for consensus models constructed using distributional data trees, highlighting their exceptional and geometrically comprehensive coverage characteristics. This fundamental trend observed in GComp is also mirrored, albeit less comprehensively due to its test-set dependency, in *FracCovered*, which, where interpretable, broadly reflects similar underlying coverage characteristics as revealed by GComp.

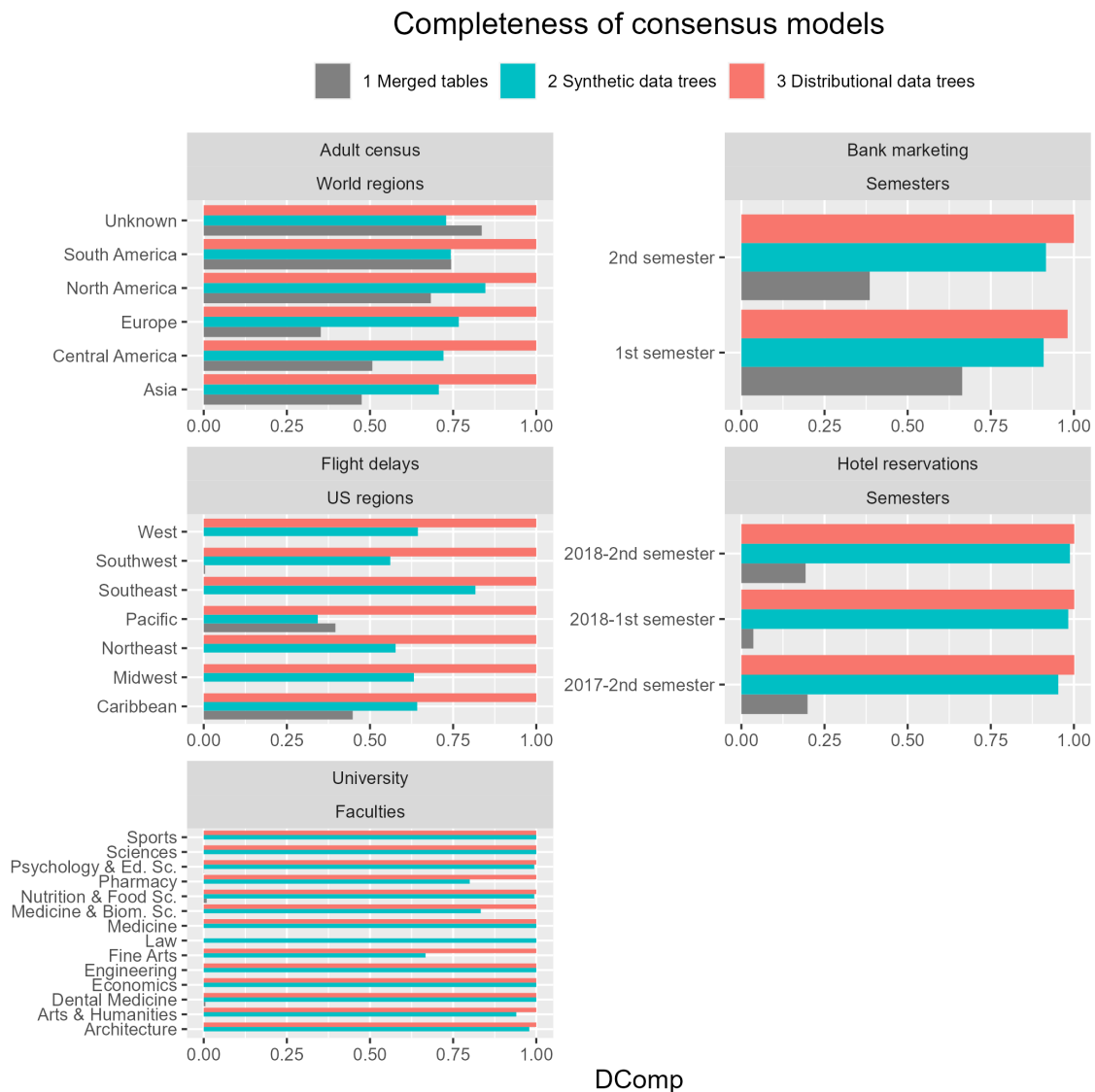


Figure 7.13: Results of the Completeness of consensus models measured with Distributional completeness

Merged table representations exhibit pronounced incompleteness primarily due to the inherent limitations of the merging process when performed with the intersection of operational models. Specifically, rule merging can inadvertently exclude certain categories or category combinations, leading to systematic gaps in coverage. Distributional data trees address this incompleteness directly by integrating distributional characteristics of independent variables during tree construction. This information allows distributional data trees to account for categories that might be absent from the merged rules, thus achieving maximum completeness. For numerical variables, even if a specific decision rule within a consensus model does not cover the entire domain of a numerical variable, the generalization capabilities of decision tree training algorithms help mitigate potential incompleteness. This is true when applied to either distributional or synthetic data, enabling extrapolation and generalization across the continuous variable domain.

7.3 Correlation analysis of evaluation metrics

The preceding sections provided independent analyses of individual evaluation criteria, namely predictive performance (including loss), fidelity, interpretability, and completeness, each offering a distinct perspective on the quality of consensus models. Given the expected interplay and potential relationships between these criteria, a correlation analysis is conducted in this section to reveal or detect such interdependencies more effectively. This approach enables a holistic and integrated understanding of how these evaluation criteria are interrelated. To this end, correlation matrices are generated and presented for each consensus model representation: Figure 7.14 specifically focuses on merged decision tables, Figure 7.15 on synthetic data tree models, and Figure 7.16 on distributional data tree models. These correlation matrices serve as visual analytical tools designed to explicitly illustrate the inter-metric relationships inherent within each distinct type of consensus model representation.

To analyse the potential linear relationships between the evaluation metrics, *Pearson correlation coefficients* (Schober et al., 2018), are employed. Pearson correlation is a widely established statistical measure specifically designed to quantify the strength and direction of linear associations between two continuous variables. In each correlation matrix, individual cells represent the Pearson correlation coefficient, denoted by R , calculated between a pair of evaluation metrics. The visual intensity of the cell color and the size of the embedded circle directly correspond to the magnitude of the correlation, providing an intuitive visual representation of relationship strength. Pearson correlation values range from -1.0 to 1.0, where -1.0 indicates a perfect negative linear correlation, 1.0 a perfect positive linear correlation, and 0 signifies the absence of any linear relationship between the metrics. This comprehensive correlation analysis encompasses all 137 consensus models developed across all datasets and merged across all stratification criteria (detailed in Table C.8), ensuring a generalizable assessment of inter-metric dependencies. The analysis reveals both consistent and representation-specific correlation patterns across the three consensus model representations, highlighting both shared and distinct inter-metric dynamics.

To explore specific inter-metric relationships highlighted by the correlation matrices, scatter plots visualizing selected metric pairs with overlaid linear regression lines are presented. Each regression line is labeled with its Pearson correlation (R) and statistical significance ($p < 0.05$ confidence). Furthermore, to visually convey the uncertainty inherent in the regression estimate, a standard confidence band is depicted around each line. This band illustrates the plausible range of the true regression line, with 95% confidence, given the observed data.

When Pearson's R exceeds 0.7, a commonly used threshold for strong linear relationships, statistical significance ($p < 0.05$) is frequently observed, especially in larger samples. This occurs because stronger correlations reflect more systematic patterns in the data, reducing the likelihood that such associations arise purely from random chance (Han et al., 2012). However, statistical significance is heavily influenced by sample size as even weak correlations (e.g., $|R| = 0.2$) can achieve significance in very large datasets, while strong correlations may fail to reach significance in small samples.

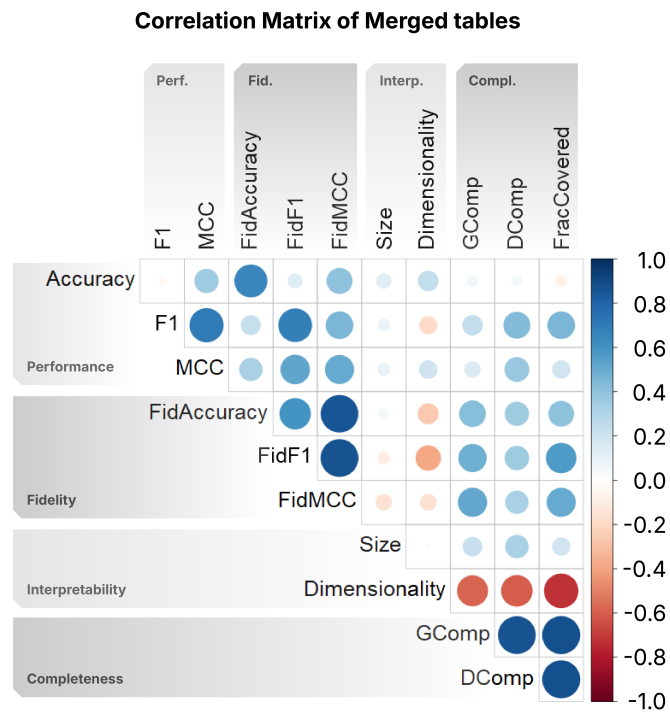


Figure 7.14: Correlation matrix of evaluation metrics for Merged tables

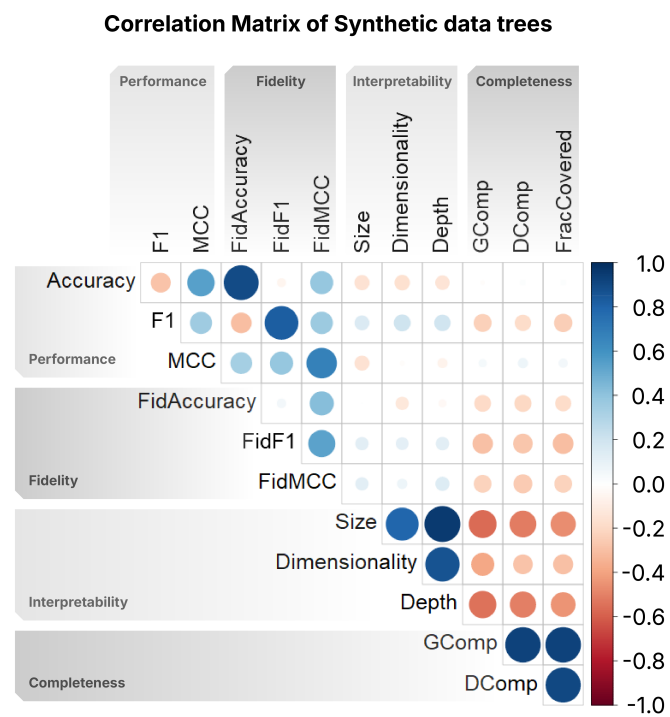


Figure 7.15: Correlation matrix of evaluation metrics for Synthetic data trees

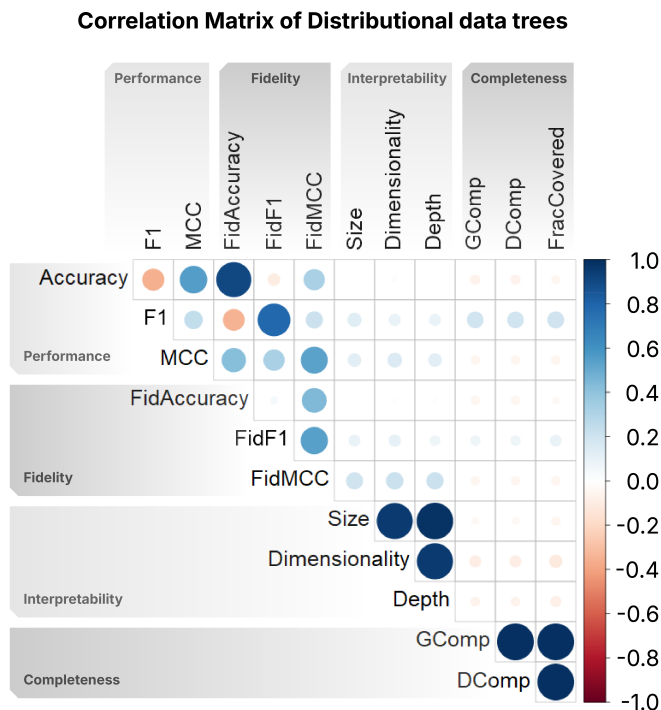


Figure 7.16: Correlation matrix of evaluation metrics for Distributional data trees

While p -values indicate whether a correlation is non-random, they do not measure the relationship's strength or practical importance. Consequently, this analysis prioritizes correlations with $|R| > 0.7$, as they represent meaningful metric relationships, irrespective of sample-driven significance. The following sections explore potential explanations for these relationships, emphasizing that statistical significance alone does not imply causation.

7.3.1 Predictive performance metrics

Analysis of the correlations presented in Figures 7.14, 7.15, and 7.16 reveals notable patterns among predictive performance metrics. A negative correlation between Accuracy and F1 is observed across all representations. This seemingly counterintuitive relationship arises from the differing sensitivities of these metrics to class imbalance. Accuracy, which is heavily influenced by the majority class, may remain high even if the model performs poorly on the minority class. In contrast, F1 directly accounts for both Precision and Recall, making it more sensitive to minority class performance, particularly when the minority class is designated as the positive class, which is the case on the datasets used for evaluation.

Conversely, Accuracy and MCC exhibit a consistently positive correlation, which aligns with expectations, as both metrics assess overall classification performance across all classes. The correlation between F1 and MCC is stronger in merged table representations. This likely arises from the fact that both F1 and MCC offer greater robustness to class imbalance than Accuracy. Consequently, they tend to align more closely in ranking model performance when class distributions

are skewed. This reinforces the rationale for prioritizing MCC as a more balanced and reliable performance metric in scenarios where class imbalance is a concern.

7.3.2 Predictive performance and Fidelity

The analysis reveals a consistent positive correlation between a consensus model’s predictive performance and its fidelity across all representations. This expected association supports the theoretical premise that models with higher fidelity, i.e., those that more accurately replicate the decision logic of the operational models, tend to achieve better predictive performance. However, the strength of this correlation varies, particularly across the different representations of consensus models. Specifically, the relationship between predictive performance and fidelity is stronger in tree-based representations (both synthetic and distributional data trees) compared to merged tables. This suggests that, for tree-based models, fidelity serves as a stronger indicator or proxy for predictive performance, whereas, in merged tables, additional factors may play a more significant role in determining performance beyond fidelity alone.

The strength of the correlation is also influenced by the choice of predictive performance metric. Figure 7.17 visually represents these correlations for MCC, showing that synthetic data trees exhibit the strongest correlation between MCC and FidMCC, while merged tables display the weakest. This variation, both across representations and between metrics, highlights the need for further investigation into the underlying factors driving these differences. Although the analysis primarily focuses on MCC, the correlation matrices reveal that the positive correlation is generally stronger when using F1 rather than MCC.

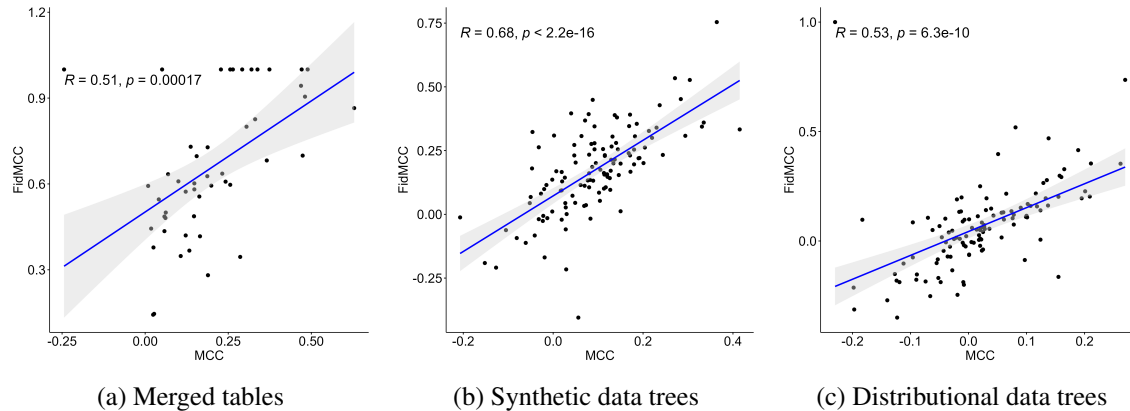


Figure 7.17: Correlations between Predictive performance (MCC) and Fidelity (FidMCC)

7.3.3 Predictive performance and Interpretability

The correlations between predictive performance and interpretability metrics, as presented in Figures 7.14, 7.15, and 7.16, are generally weak across all representations. This suggests that, when considered in isolation, these two evaluation criteria exhibit limited direct linear association. However, the direction and subtle variations in these weak correlations are representation-dependent.

A weak positive correlation is observed for merged tables, indicating a slight tendency for more complex merged tables (i.e., those with lower interpretability) to achieve marginally better predictive performance. Conversely, synthetic data tree representations show a weak negative correlation, suggesting that slightly more interpretable synthetic trees may exhibit marginally better predictive performance. For distributional data trees, a weak positive correlation is again observed, similar to merged tables, though the magnitude of this association is even weaker. These generally weak and representation-dependent correlations highlight the complex and non-linear nature of the relationship between predictive performance and interpretability, suggesting that a simple linear trade-off may not fully capture their intricate interplay.

7.3.4 Predictive performance and Completeness

Figure 7.18 reveals a slight positive correlation between predictive performance (MCC) and completeness (Distributional completeness) for merged tables, suggesting that merged tables with higher completeness (i.e., better decision space coverage) tend to exhibit marginally improved predictive performance. This relationship implies that completeness may play a role in enhancing predictive capabilities, though its effect appears limited.

In contrast, for tree-based representations (both synthetic and distributional data trees), no discernible correlation between the criteria is found. This lack of correlation suggests that, within the completeness levels achieved by tree-based models, completeness itself is not a primary factor driving predictive performance variation. For merged tables, however, the slight positive correlation may indicate that attaining a minimal level of completeness is somewhat necessary for better predictive performance, although completeness alone remains an insufficient condition for high predictive performance.

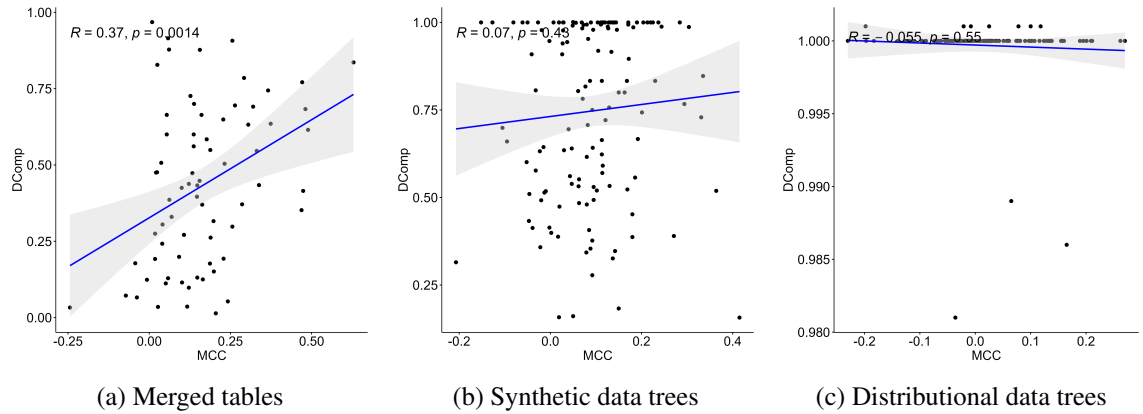


Figure 7.18: Correlations between Predictive performance (MCC) and Completeness (DComp)

7.3.5 Fidelity metrics

Figures 7.14, 7.15, and 7.16 show that correlations among fidelity metrics (FidAccuracy, FidF1, and FidMCC), as expected, are consistently positive across all representations, given that these

metrics capture different aspects of inter-model agreement. However, the strength and pattern of these correlations vary across representations, reflecting differences in how fidelity is observed within each model type.

In merged table representations, these correlations tend to be stronger and more uniformly positive, indicating a tighter alignment and potential redundancy among fidelity measures. In tree-based representations, the correlation patterns among fidelity metrics closely resemble those observed for predictive performance metrics, for instance, a negative correlation between FidAccuracy and FidF1 and a positive correlation between FidAccuracy and FidMCC. This suggests that, for tree-based models, fidelity evaluation shares similarities with predictive performance evaluation, reinforcing the intricate relationship between these two evaluation criteria.

7.3.6 Fidelity and Interpretability

The analysis reveals a consistent lack of discernible correlation between fidelity and interpretability metrics across all representations, as illustrated in Figure 7.19 (for FidMCC and Dimensionality). This absence of correlation indicates that, within the context of this consensus model evaluation, fidelity to operational models and the interpretability of consensus models, measured through complexity metrics, are largely independent criteria of model quality, as confirmed by the correlation matrices for metrics beyond dimensionality.

This finding suggests that a model can maintain a high level of fidelity to the predictions of a complex operational model without being inherently more or less interpretable. Conversely, a model that exhibits high interpretability does not guarantee fidelity to the operational model's predictions, highlighting the need to evaluate these qualities separately in the context of model evaluation.

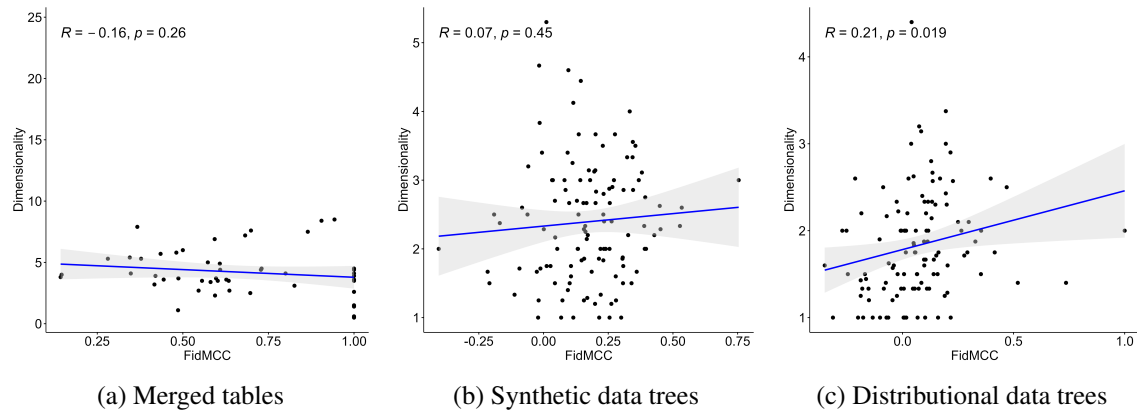


Figure 7.19: Correlations between Fidelity (FidMCC) and Interpretability (Dimensionality)

7.3.7 Fidelity and Completeness

As depicted in Figure 7.20 (for FidMCC and Distributional Completeness) a slight positive correlation is observed between fidelity and completeness specifically in merged table representations.

This suggests a subtle tendency for merged tables that exhibit higher fidelity to operational models to have slightly higher completeness.

However, this slight positive correlation becomes essentially non-existent in tree-based representations. This representation-specific correlation pattern may indicate that the trade-offs between fidelity and completeness are more pronounced or manifested differently in merged tables compared to tree-based representations. For tree-based models, fidelity and completeness appear to operate largely independently.

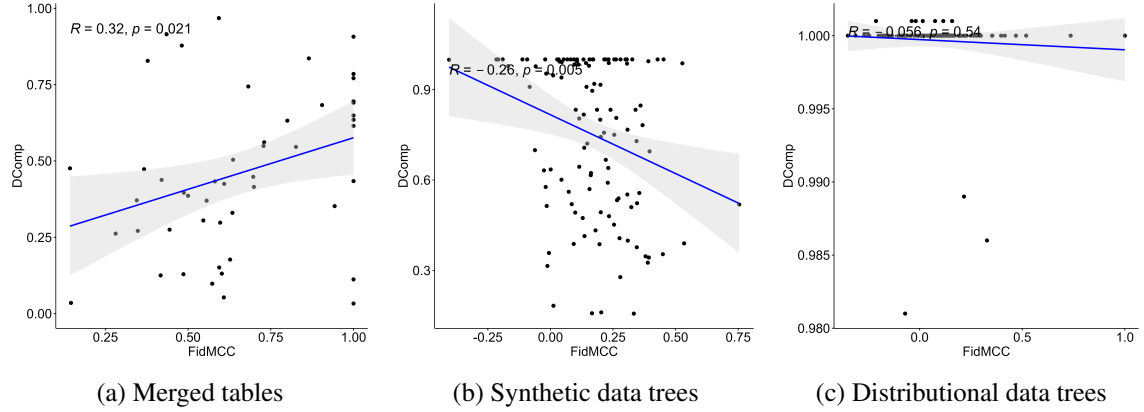


Figure 7.20: Correlations between Fidelity (FidMCC) and Completeness (DComp)

7.3.8 Interpretability metrics

Figures 7.14, 7.15, and 7.16 illustrate a strong positive correlation between interpretability metrics, specifically Size, Dimensionality, and Depth, in tree-based representations, while such correlations are non-existent in merged tables. As shown in Figure 7.21 (depicting Size and Dimensionality), this positive correlation suggests a consistent alignment among these complexity measures: larger distributional trees typically use more variables and exhibit greater depth. In contrast, the absence of a similar correlation in merged table representations may reflect a more intricate or less predictable relationship between various aspects of structural complexity in these models.

The observed lack of strong positive correlation among interpretability metrics in merged tables, compared to stronger correlations in tree-based representations, are likely to stem from their construction methods. Merged tables are created through a process that intersects rule sets and resolves conflicts among rules. Consequently, the complexity metrics can become decoupled. For example, a merged table's number of rules may increase due to intersecting rules from various operational models without a proportional increase in distinct variables. Therefore, Dimensionality in merged tables is often driven by the diversity of variables from contributing operational models rather than a structured feature selection process.

Tree-based models are constructed using algorithms that iteratively partition data by selecting splits that minimize impurity, which measures the heterogeneity of class labels within a node (Han et al., 2012). By reducing impurity, the algorithm prioritizes splits that produce increasingly homogeneous subsets in terms of class distribution. This process inherently relates to complexity

metrics such as Size (total number of nodes or rules), Depth (levels in the tree), and Dimensionality (number of distinct variables used). As each split decision introduces hierarchical dependencies, deeper trees (increased Depth) generally require more splits (larger Size) and may involve additional variables (higher Dimensionality). However, the repeated use of variables can limit dimensionality growth, while pruning techniques may reduce both tree size and depth.

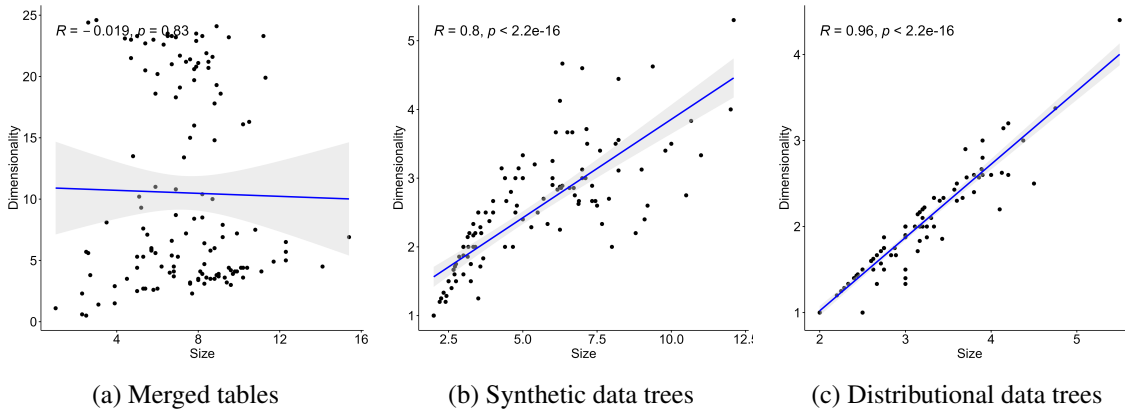


Figure 7.21: Correlations between Interpretability metrics (Size and Dimensionality)

7.3.9 Interpretability and Completeness

Figures 7.14, 7.15, and 7.16 shows the expected inverse relationship between model interpretability and completeness, suggesting that more complex models are generally anticipated to exhibit lower completeness. However, in the case of merged tables, this anticipated trend is not observed; counterintuitively, an increase in the number of rules tends to correlate with better (higher) coverage of the decision space. This phenomenon likely arises because a more rules directly implies a large portion of the decision space covered. Conversely, a subtle negative trend emerges in synthetic data trees, suggesting that larger trees may exhibit marginally lower completeness. This indicates a slight trade-off between the number of rules and coverage in the generation of synthetic trees. In particular, this expected trade-off is absent in distributional data trees, where there is essentially no correlation between Size and Distributional Completeness (Figure 7.22).

Furthermore, the correlation between Dimensionality and Completeness is found to be slightly stronger and negative specifically in merged tables. This trend indicates that increasing the number of variables in merged tables leads to coverage gaps, potentially due to increased fragmentation of the decision space. In contrast, this negative trend is largely absent in tree-based representations (both synthetic and distributional data trees). As illustrated in Figure 7.23, showing the correlation between Dimensionality and Distributional Completeness, it appears that for tree-based models, dimensionality and completeness are less directly coupled compared to their counterparts in merged tables.

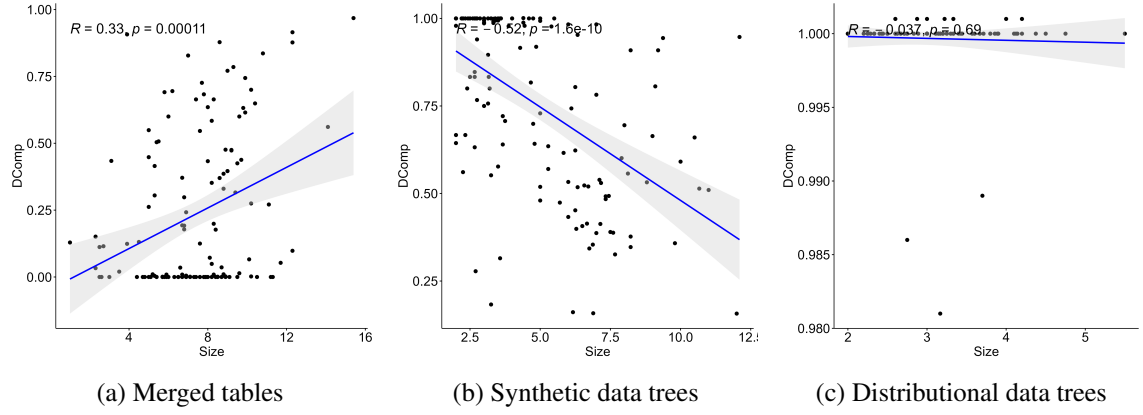


Figure 7.22: Correlations between Interpretability (Size) and Completeness (DComp)

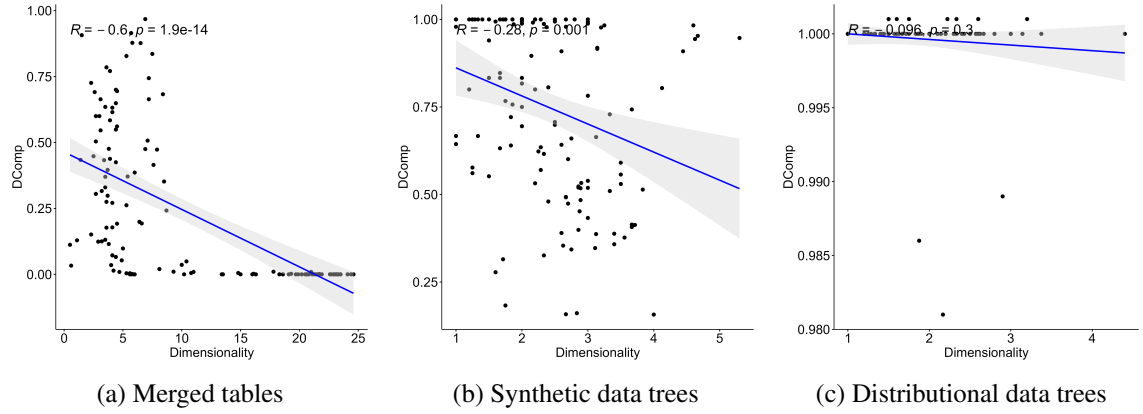


Figure 7.23: Correlations between Interpretability (Dimensionality) and Completeness (DComp)

7.3.10 Completeness metrics

As anticipated, completeness metrics demonstrate a strong positive correlation among themselves, particularly pronounced in distributional data tree representations. This correlation, illustrated in Figure 7.24 (showing Geometrical and Distributional Completeness), is expected due to the inherent conceptual alignment of these metrics. Both Geometrical Completeness and Distributional Completeness aim to quantify the extent of decision space coverage, though they use different weighting schemes (uniform versus data-density weighted).

Notably, the strong positive correlation observed in distributional data trees suggests that improvements in GComp (uniform coverage) are effectively mirrored by improvements in DComp (data-density weighted coverage). This finding points to a consistent and robust completeness profile for distributional data tree representations, underscoring their effectiveness in comprehensively covering the decision space.

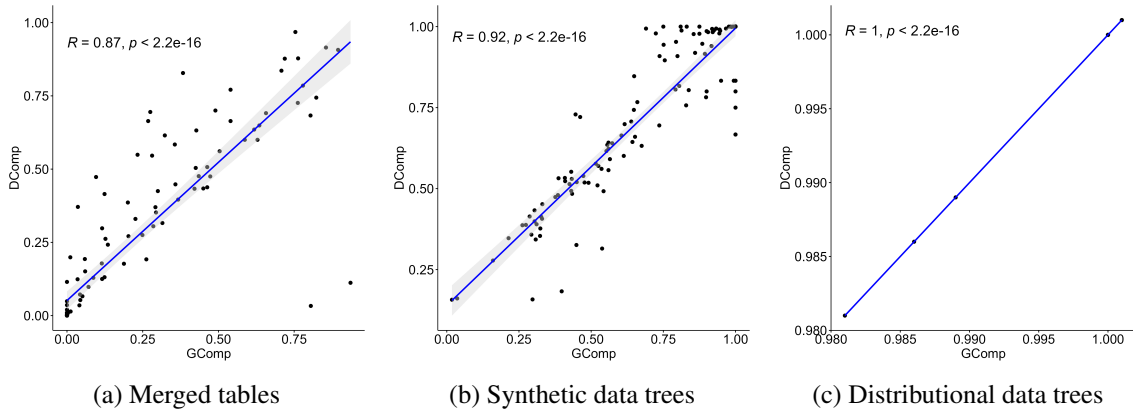


Figure 7.24: Correlations between Completeness metrics (GComp and DComp)

7.4 Key trends and trade-offs from consensus model evaluation

The evaluation of consensus models, using both isolated criteria and correlation analysis, uncovers distinct trends and trade-offs in predictive performance, fidelity, interpretability, and completeness across various representations. While some findings corroborate theoretical expectations, others reveal unexpected behaviors that warrant further investigation.

Fidelity exhibits a clear positive correlation with predictive performance. A significant reduction in both fidelity and predictive accuracy occurs when transitioning from merged tables to tree-based representations. However, assessing fidelity in merged tables is often unreliable due to near-zero geometrical completeness, limiting the validity of comparisons. Additionally, no strong correlation is observed between predictive performance and completeness, indicating that completeness is not a primary determinant of predictive performance within the evaluated conditions.

The decline in fidelity when shifting from tables to trees mirrors the trend observed for predictive performance. However, fidelity measurements remain constrained in cases where decision space coverage is low. Correlation analysis does not indicate a significant relationship between fidelity and model complexity, nor does it establish any clear link between fidelity and completeness across different representations.

Interpretability significantly improves in tree-based representations compared to merged tables. In tree-based models, there is a strong positive correlation among interpretability metrics, indicating a consistent relationship in which increased Dimensionality correlates with higher Size and greater Depth. In contrast, merged tables do not follow this trend, as they exhibit a lack of strong correlation between Size and Dimensionality. This divergence highlights the fundamental structural differences in the complexity of different consensus model representations.

When examining the relationship between interpretability and completeness, a subtle trend emerges, suggesting a potential inverse relationship, where increased complexity may be linked to reduced completeness, in synthetic data tree representations. However, this relationship is not consistently observed across other consensus model representations. Therefore, while a possible trade-off is suggested for synthetic data trees, a generalized inverse relationship between inter-

pretability and completeness cannot be definitively established across all consensus model representations.

Consensus model completeness shows a significant improvement in tree-based representations compared to merged table representations. In particular, distributional data trees consistently approach complete coverage of the decision space. The two distinct completeness metrics used in this evaluation, Geometrical Completeness and Distributional Completeness, exhibit a robust positive correlation across all consensus model representations, with the latter consistently achieving higher scores than the former across the evaluated models. This suggests that tree-based consensus models, particularly distributional data trees, effectively prioritize and cover areas of high data density within the decision space, even if they do not ensure uniform coverage. This trend highlights the inherently data-driven characteristics of tree-based consensus model representations, most notably in the distributional data trees.

7.5 Summary

The comprehensive results from the empirical evaluation of the *Inmplode* framework clearly demonstrate both the potential and limitations of consensus models, particularly in complex, real-world decision-making scenarios. The direct use of consensus models as substitutes for high-performance operational models faces ongoing challenges. These challenges include consistently achieving acceptable levels of predictive performance and fidelity, especially when high predictive performance is required.

However, the findings highlight the significant value of consensus models as tools for enhanced explanation, knowledge discovery, and organizational knowledge consolidation. By synthesizing and aggregating decision-making information across various business units, consensus models offer more interpretable and structurally comprehensive representations of collective decision logic than individual operational models. This improved interpretability and comprehensiveness enable consensus models to foster a deeper organizational understanding of complex phenomena. They also facilitate knowledge sharing across silos and promote more robust, holistically informed knowledge consolidation within data-driven organizations.

Chapter 8

Conclusions

8.1 Summary and contributions

Interpretability, although lacking a universally accepted definition, is generally perceived as the ability of models to be understandable to humans. This multifaceted concept encompasses both local interpretability, focusing on the explanation of individual predictions, and global interpretability, which pertains to understanding the overarching logic and reasoning embedded within the model's predictive behavior. Decision trees, a widely used rule-based model, are considered interpretable due to their transparent structure. However, in decentralized environments, where multiple models provide localized insights, managing interpretability becomes challenging. The individual models may offer valuable information but fail to provide a cohesive understanding of the phenomenon from a broader organizational perspective. This fragmentation of knowledge across disparate models hinders the ability to synthesize a unified, strategic understanding.

In decentralized contexts, where different business units develop their own operational models, maintaining a global understanding becomes increasingly difficult. For example, a chain of stores may use decision trees to predict profits at each location, while a university may apply them to assess student performance in individual courses. As the number of operational models grows, it becomes harder to maintain interpretability across the organization. Therefore, global interpretability can be reframed as the challenge of synthesizing knowledge dispersed across multiple related operational models. Driven by the hypothesis that machine learning methods can address this challenge, this thesis proposes an approach to merge multiple operational models into consensus models, which aggregate localized knowledge into a broader, more strategic perspective. This hypothesis further underscores the paramount importance of representativeness as a key desideratum for consensus models. This translates as the degree to which they faithfully capture and synthesize the distributed knowledge and decision logic embedded within the constituent operational models they are intended to represent. To this end, a number of novel approaches and methods have been developed and evaluated, forming the core contributions of this research.

The primary contribution of this research is the formalization of a novel analytical framework for merging rule-based models. This framework, named *Inmplode*, is grounded in a comprehensive

Conclusions

analysis of existing model merging techniques and the explicit identification of their underlying common principles. *Inmplode*, presented in Chapter 4, provides a flexible and extensible framework for model merging, characterized by a generic, adaptable workflow and a modular design incorporating customizable components. This framework is explicitly designed to address a range of inherent challenges in model merging, including: providing a unified and consistent representation for diverse rule-based model formats, effectively combining and integrating decision rules extracted from disparate operational models, systematically resolving logical conflicts and redundancies arising from overlapping rule sets, and implementing rule simplification and abstraction techniques to enhance the overall interpretability of the resulting consensus model. The practical applicability of *Inmplode* is evaluated through a detailed case study implementation within the University of Porto, showcasing its efficacy in a real-world organizational context. The intricate specificities and algorithmic details of the core model merging process are further elaborated in Chapter 5.

The second main contribution addresses the frequent incompleteness observed in consensus models, which stems from the merging process. This incompleteness manifests as uncovered examples, which are data points for which the model lacks sufficient decision space coverage to make a prediction. Such gaps can defeat the consensus model's purpose. This research explored novel strategies to restore completeness when representing these models as decision trees, presented in Chapter 6. The first approach, named *Synthetic data trees*, involved generating synthetic datasets from the merged rules and applying standard decision tree induction algorithms. While effective to some extent, this method faced challenges in achieving full coverage of the decision space. Therefore, the study introduced another novel approach based on the Symbolic Data paradigm, resulting in the development of another method, named *Distributional data trees*. This method represents rules as distributional data, allowing the use of specialized decision tree induction algorithms designed for such representations. The distributional data trees method proved particularly effective in generating complete consensus models capable of providing predictions across the entire decision space.

While addressing the challenge of regaining completeness in consensus models, two additional supplementary contributions emerged. The first is the development of novel metrics to evaluate model completeness without relying on specific test sets, which may not be available in the context of consensus models. To this end, two metrics are introduced: *Geometrical completeness* and *Distributional completeness*, detailed in Appendix B. The former evaluates the coverage of the decision space by measuring the proportion of the space for which a model can provide predictions. The latter evaluates coverage relative to data concentration. Both metrics enable the evaluation of a model's completeness independently of test set availability. In the evaluation of consensus model completeness, distributional completeness consistently demonstrates that these models can effectively cover regions of high data concentration, despite not fully covering the entire decision space.

The second supplementary contribution arose from the need to calculate distributional completeness, which requires density estimation for high-dimensional data. Standard density esti-

mation methods, such as Kernel Density Estimation, become computationally infeasible as the number of variables increases. To address these challenges, within this research a novel density estimation method based on multivariate histograms was developed. Unlike traditional approaches, this method is specifically designed to handle high-dimensional data while accommodating categorical variables and missing data, as detailed in Appendix A. The effectiveness of this method was evaluated across multiple datasets. The results demonstrated its ability to estimate data density in high-dimensional spaces, making it particularly suitable for computing distributional completeness.

8.2 Evaluation of the hypothesis

This thesis set out to evaluate the hypothesis that consensus models can effectively summarize and represent the knowledge contained within multiple rule-based operational models. The results generally support this hypothesis. Using *Inmplode*, consensus models were developed and evaluated based on multiple criteria, including predictive performance, fidelity, interpretability, and completeness, across five distinct datasets. The evaluation distinguishes between three representations of consensus models: merged decision tables (resulting from merging operational models), and decision tree representations (synthetic data trees and distributional data trees), which were developed both to restore completeness and to present the consensus models in same format as the operational models.

While decision trees show a decline in predictive performance compared to merged decision tables, this trade-off is outweighed by their enhanced ability to capture valuable insights across multiple models. This is paramount for decentralized and complex operational environments. Fidelity analyses confirm this balance, showing that decision trees, though less precise in replicating individual operational model outputs, still maintain enough alignment to aggregate the knowledge of operational models effectively.

One of the key strengths of consensus models is their interpretability. Distributional data trees simplify complexity, distilling operational logic into fewer rules and variables. Their effectiveness is matched by greater completeness, as these models generalize well across diverse decision spaces, addressing coverage gaps found in other representations. By integrating variable distributions into their structure, distributional data trees are able to regain completeness that may be lost in the process of merging operational models.

In summary, this research confirms the viability of consensus models, particularly distributional data trees, for synthesizing operational model knowledge into interpretable and comprehensive representations. Despite challenges in fidelity and predictive performance, their gains in interpretability and completeness justify their use in decentralized operations. These findings validate the initial hypothesis and highlight the need for further refinement, especially in improving fidelity. This work lays the groundwork for more effective model merging and decision support strategies, advancing the field and opening new avenues for exploration.

8.3 Future work

A few potential directions for future research emerge from this work, which could expand the scope and applicability of the *Inmplode* framework while further refining the model merging process:

- Framework versatility – While the *Inmplode* framework offers a flexible approach to model merging, further investigation is needed to explore alternative merging heuristics. Strategies based on rule complexity, error rates, or other performance metrics could enhance the merging process. Additionally, research should focus on improving the simplification of merged models, ensuring they retain the capacity to represent the original knowledge accurately.
- Impact of merging order – The sequence in which models are merged can significantly influence the final consensus model. As the merging process is inherently sequential, examining how the order of merging affects the complexity, fidelity, and interpretability of the resulting model would be valuable. Understanding the impact of merging order could lead to more effective model consolidation strategies and improve the generalizability of findings across diverse datasets.
- Automatic stratification – This research used a business-concept-driven approach to model stratification, where operational models are grouped based on their relevance to specific decision-making processes. Future studies could explore automatic stratification methods, using meta-information from models or clustering algorithms to create more optimal groupings. This could improve consensus model generation by uncovering relationships between models that are not immediately apparent.
- Incorporating domain knowledge – Another promising direction for future research is the integration of domain-specific knowledge into the model merging process. By incorporating expert input to guide the merging of rules, it may be possible to enhance both the interpretability and predictive performance of consensus models. This approach could be particularly beneficial in fields where domain knowledge is critical for model design and decision-making.

Appendix A

Density Estimation in High Dimensional Spaces: a Multivariate Histogram Approach

To evaluate the completeness of consensus models within the *Inmplode* framework, Appendix B introduces the metric of distributional completeness. This metric is based on the model’s coverage of data concentration within the decision space, requiring a density estimator. However, existing methods are often limited to low-dimensional spaces (typically up to six dimensions). Therefore, this appendix proposes a novel density estimation method based on multivariate histograms. This method is specifically designed for scalability to higher dimensions and can process both numerical and categorical variables, as well as missing data. The method’s capabilities are illustrated through its application to a diverse set of datasets, with results presented at the end of this appendix.

A.1 Density estimation and related concepts

Density is a scalar quantity that measures the concentration of a phenomenon in a unit of space. Typical examples are the population density of a country, i.e., how many people on average are in a square kilometer (unit of area). If the phenomenon is the concentration of a gas, density refers to the average number of particles contained in a cubic meter (unit of volume). When the concept is applied to the data generated by a phenomenon, density is the average number of observations in a unit of hypervolume (a generalization of length, area, and volume to any number of dimensions).

In statistics, density is used to calculate probabilities, a fundamental concept for understanding the likelihood of events occurring. Its importance extends beyond this core application, finding significant use in machine learning as well during the early stages of data analysis, known as *Exploratory Data Analysis* (EDA), density estimation helps to reveal important characteristics of the data. This includes identifying properties such as skewness, multimodality, and the presence

of anomalies. Density analysis can also reveal the need to collect more data (Silverman, 1986). Several applications use density in various research areas, described in Section A.1.2.

Often, a dataset is a population sample, making it possible to estimate the population's actual density by calculating the dataset's density. Before presenting the formal definition of the density estimation function, it is important to revisit a few concepts. A *random variable* formalizes a quantity or object that depends on random events or experiments with an unpredictable outcome. If the possible results are finite and can be counted, the variable is *discrete* (e.g., the 52 possibilities when drawing a card from a deck). However, if the outcomes are endless and thus impossible to count, the variable is *continuous* (e.g., the infinite range of actual values between two integers). Whether discrete or continuous, some values occur more frequently than others. Such behaviors can be described by a *probability distribution*, a function specifying how likely a random variable is to take each possible value. In other words, it expresses the *probability* of the different outcomes of an event or experiment, ranging from zero (impossibility) to one (absolute certainty).

Probability distributions are treated differently for discrete and continuous variables. For discrete variables, it is used a *probability mass function*, which provides the probability of each specific value. However, for continuous variables, as there is an infinity of possible values, the probability of taking a specific value is zero. For this reason, a *probability density function* is used, oftentimes referred to simply as *density*. Its graphical depiction is a curve, and a probability is calculated as the area under the curve that corresponds to a range of two specific values. Formally, density is a non-negative function $p(x)$ that describes the distribution of a continuous random variable x . An example is a Normal distribution (Rencher and Schaalje, 2008), $N(\mu, \sigma^2)$, with parameters *mean*, denoted as μ , and *variance*, denoted as σ^2 (Equation A.1). From $p(x)$, the probability P that a value of the variable x is within an interval $[a; b]$ can be obtained by integration (Equation A.2).

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \quad (\text{A.1}) \quad P(a \leq x \leq b) = \int_a^b p(x)dx \quad (\text{A.2})$$

Density is usually estimated, as observations in a dataset are random samples of a larger set or population. *Density estimation* is the process of using a sample dataset to find a *density estimator*, $\hat{p}(x)$, a function that provides an approximation of the unknown density, $p(x)$, of a population (Chen, 2017). Density can be estimated with parametric and non-parametric approaches, described next.

A.1.1 Approaches to density estimation

A.1.1.1 Parametric approaches

Parametric approaches to estimate density are used when a preliminary analysis of the dataset reveals that it follows a common, well-known distribution (e.g., Normal, Gamma) or this is assumed, even without preliminary analysis. Density is estimated by tuning the known distribution parameters to fit them as closely as possible to the data distribution. The main advantage of this process

is the simplicity in which the density estimate can be obtained and also the possibility of understanding the parameters (Scott, 2004). However, it is also limited to scenarios in which the data distribution naturally corresponds to a known distribution, which is less common in exploratory analysis of independent random variables. For instance, assuming a dataset with n examples that can be fitted to a Normal distribution (Equation A.1), the density estimator (Equation A.3) uses the estimates for its parameters. While there are many approaches for estimating the parameters (e.g., method of moments or maximum likelihood method), a very simple method is estimating the sample mean, $\hat{\mu}_n$ (Equation A.4) and the sample variance, $\hat{\sigma}_n^2$ (Equation A.5) from the observations (X_i) of a dataset. Other data distributions have different parameters, which also have to be estimated.

$$\hat{p}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma_n^2}(x-\hat{\mu}_n)^2} \quad (\text{A.3})$$

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n X_i \quad (\text{A.4})$$

$$\hat{\sigma}_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \hat{\mu}_n)^2 \quad (\text{A.5})$$

A.1.1.2 Non-parametric approaches

Non-parametric approaches to estimate density are used when the data distribution is unknown. This typically involves either devising a continuous function for a more accurate approximation or aggregating data in a discrete function for a rougher estimate.

Continuous function density estimators estimate density by devising a continuous function from the data distribution, similar to training a prediction model via regression. This approach offers the advantage of flexibility, allowing density estimation for any distribution regardless of its shape or whether it is common. However, there are also limitations to its application. As the dimensionality of the dataset increases, density estimation becomes harder with escalating computational costs, both in available memory and elapsed time. Typically, implementations deal with two dimensions such as *GenKern* (Lucy and Aykroyd, 2010) and *KernSmooth* (Wand and Ripley, 2010), among others. Very few accommodate more, as is the case with *sm* (three dimensions) (Duong, 2011) and *ks* (Bowman and Azzalini, 2010) (six dimensions).

While there is a wide variety of algorithms, *Kernel Density Estimators* (KDE) (Węglarczyk, 2018) stand out as one of the most commonly used. Formally, in KDE, the parameters include the *kernel function* (K), typically a smooth, symmetric function, and the *smoothing bandwidth* (h), a positive value controlling the degree of smoothing (Equation A.6). The method takes a dataset as input and replaces each observation (e_i) with a small density agglomeration shaped according to the kernel function. The density function then consists of the sum of all the agglomerations; therefore, each example influences any given data point (x) in the density estimate.

Many kernel functions are available, producing various shapes such as Gaussian, Epanechnikov, or Biweight (Węglarczyk, 2018). The smoothing bandwidth scales the agglomeration. A lower bandwidth implies that only examples close to x carry significant weight, resulting in a more

irregularly shaped estimate. Conversely, a higher bandwidth leads to a more flattened kernel, allowing contributions from examples farther away from x to influence the estimate (Chen, 2017).

$$\hat{p}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - e_i}{h}\right) \quad (\text{A.6})$$

Discrete function density estimators, often referred to as *histogram-based approaches*, estimate density by counting the frequency of similar observations and grouping them into bins of equal or variable width. Due to their simplicity, histograms are calculated quickly and with low computational costs without being penalized by increasing dimensionality, even for large datasets. However, this performance is obtained as a trade-off with the quality of the underlying density estimator. A count-based method generally provides a rough estimate compared to continuous function density estimators. Moreover, when discrete bins replace a continuous function, the origin and width of these bins can significantly alter the density estimate, making the method highly sensitive to the selection of these parameters. There are, however, a few enhancements like the *Average Shifted Histogram* (ASH) to decrease the sensitivity to the choice of origin (Scott, 1985).

Formally, a *histogram* is a function that partitions the continuous domain of a numerical variable (x) into a finite set $\mathcal{B}$ of contiguous intervals, also known as *bins*, denoted by b_k (Equation A.7). Each bin has a specific width (w_k), and the histogram counts the number of data points (denoted by v_k) within a dataset that falls into each bin (Equation A.8, using an indicator function 4.9). The total number of bins is denoted by K .

$$\mathcal{B} = \{b_1, \dots, b_K\} \quad (\text{A.7}) \quad v_k = \sum_{i=1}^n I[x_i \in b_k] \quad (\text{A.8})$$

Considering a numerical variable x with the interval $[a_1; a_2]$ as domain, the partition into bins is given by Equation A.9, where b_k denotes each bin.

$$b_k = \begin{cases} [a_1; a_1 + w_1] & k = 1 \\ (a_1 + \sum_{i=1}^{k-1} w_i; a_1 + \sum_{i=1}^k w_i] & 1 < k < K \\ (a_1 + \sum_{i=1}^{k-1} w_i; a_2] & k = K \end{cases} \quad (\text{A.9})$$

The histogram as a density estimator (Scott, 2010) is defined as the relative frequency (v_k/n , n being the total number of examples in the dataset) of a specific bin (b_k) by its width (w_k) (Equation A.10). A histogram assumes a uniform distribution of data points within each bin. Consequently, the density estimate is constant for all data points within a single bin.

$$\hat{p}(x) = \frac{v_k}{n \times w_k}, x \in b_k \quad (\text{A.10})$$

While histograms are traditionally used for continuous variables, they can be adapted for the frequency distribution of categorical variables by numeric encoding, where each category is assigned a unique numerical value.

A.1.2 Applications in research

Beyond probability calculations, density estimation methods have been applied across diverse fields, yielding significant insights. In medical research, for instance, analysis of degranulated mast cell counts in 95 infants who died from unexplained causes, compared to a control group of 76 infants with known causes, revealed a high concentration of these cells in 25-33% of unexplained deaths, prompting further clinical investigation (Carpenter and Emery, 1974). In biology, density estimation applied to directional data from 76 turtles released into the ocean revealed a bimodal distribution; most turtles swam towards a 60° direction, while a smaller group moved in the opposite direction, raising questions about the underlying behavioral mechanisms (Mardia, 1975). In materials engineering, analysis of height measurements taken at 15000 points on steel surfaces resulted in a unimodal distribution whose extended tails indicated wear-prone areas and potential leak points, guiding improvements in fabrication processes (Adler et al., 1981).

Density estimation has also been used in the social and environmental sciences. In social science, Kernel density estimation was used to evaluate the distribution of income across different demographic groups over time (Piketty and Saez, 2003). In environmental science, Density estimation methods were used to model species distribution and abundance in ecological studies (Elith et al., 2011). This approach helps predict the occurrence of plant species across various environmental gradients, aiding in conservation efforts. More recently, density estimation has been used in machine learning. For example, it has been used in deep autoregressive models to generate natural images with minimal data, enabling effective few-shot density estimation through the incorporation of neural attention and meta-learning (Reed et al., 2018).

A.1.3 Example: Density of the Old Faithful dataset

The Old Faithful Geyser dataset, available from Kaggle (Kaggle, 2024), can be used as an example for illustrating density visualization. This dataset contains 272 observations of two variables: eruption time (in minutes) and waiting time for the next eruption (also in minutes). Figure A.1 depicts these variables in a scatter plot, offering an initial glimpse into their relationship.

Density estimation techniques can be employed to explore the data's distribution further. Figures A.2 and A.3 present univariate density estimations for each variable separately. These visualizations, presented both as histograms and density curves, reveal that both eruption time and waiting time exhibit bimodal distributions. This means that the data clusters around two distinct peaks, suggesting potential patterns or underlying factors influencing the geyser's behavior.

Moving beyond single variables, Figures A.4, A.5, and A.6 showcase bivariate density estimations, considering both variables simultaneously. Figure A.4 presents a 2-dimensional histogram with a 10×10 grid, where lighter rectangles indicate areas of higher density. This visualization enables the observation of the joint distribution of eruption time and waiting time, revealing regions where data points are more likely to occur. Figure A.5 refines this visualization with a 30×30 grid, offering a more granular view of the density distribution. This visualization, featuring smaller rectangles and more areas of zero density, provides a more precise representation of

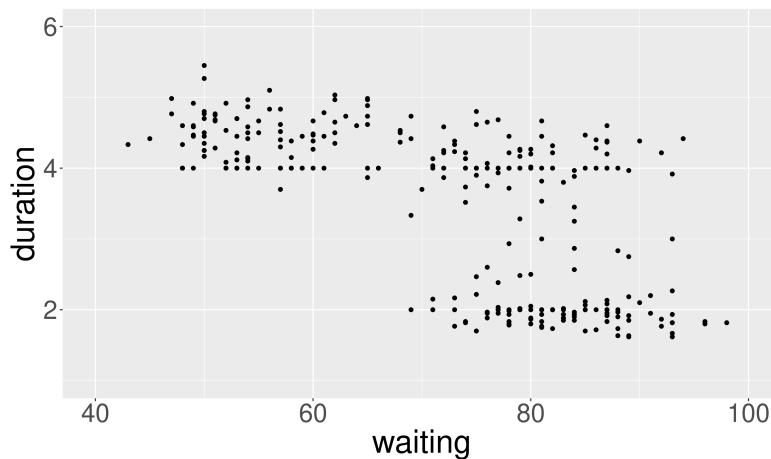


Figure A.1: Old Faithful data

the data's spread. Finally, Figure A.6 depicts a density surface obtained through a non-parametric approach. This smooth surface visualization provides a continuous representation of the joint density, further identifying areas of high and low probability for specific combinations of eruption time and waiting time.

A.2 A multivariate histogram-based method for density estimation

Parametric density estimation methods are constrained by the need for data to conform to a known distribution, limiting their applicability. While non-parametric approaches offer broader scope and are generally more accurate for density estimation, their computational complexity increases significantly with dataset dimensionality. Kernel Density Estimators (KDEs), a prime example of non-parametric methods, become computationally prohibitive even in moderately-sized datasets, typically beyond six dimensions. Although histograms provide rougher density estimates, they remain the only viable option for higher-dimensional data. However, existing histogram implementations still struggle with reasonably sized datasets and are further restricted by data type;

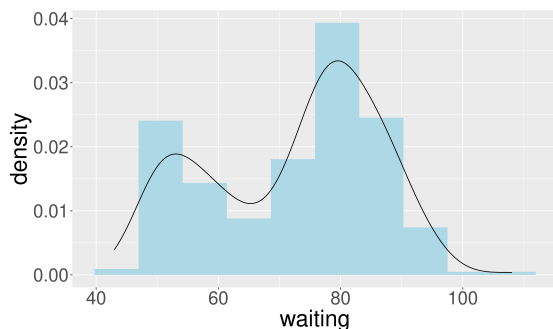


Figure A.2: Density of 'waiting'

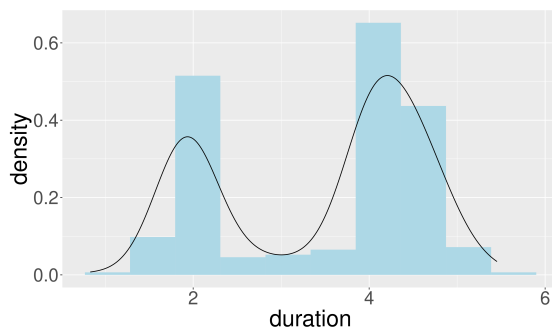


Figure A.3: Density of 'duration'

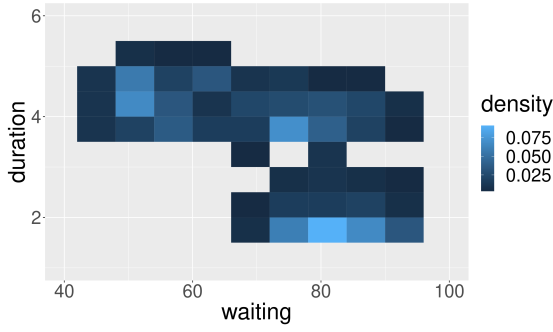


Figure A.4: 10×10 histogram

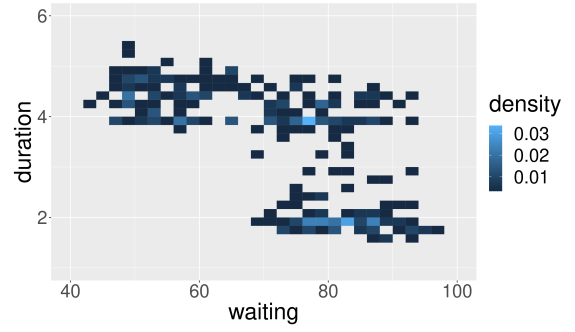


Figure A.5: 30×30 histogram

they can only represent the frequency of continuous numerical data and cannot handle categorical variables. Moreover, current histogram methods do not naturally accommodate missing data, requiring preprocessing steps such as imputation or removal of incomplete observations. These limitations motivate the development of a novel density estimation method specifically designed to address the challenges of measuring incompleteness when developing consensus models within the *Inmplode* framework.

The proposed density estimation method relies on a two-step approach: first, a multivariate histogram is constructed to represent the data distribution. Subsequently, this histogram is used as the basis for a density estimator. Figure A.7 presents an overview of the approach. A 1-dimensional histogram partitions the domain of a variable into bins bounded by breakpoints (defining a 1-dimensional grid). The combination of the domains of all variables in a dataset represents a *hyperspace*. Therefore, a multivariate histogram partitions the hyperspace into a *hypergrid* bounded by edges (defining a d -dimensional grid). Each partition is defined as a *hyperrectangle*, i.e., a subspace within the hyperspace. Subsequently, a multivariate histogram with relative frequencies for each hyperrectangle (Γ_f) is calculated. This leads to the creation of the density estimator ($\Gamma_{\hat{\rho}}$) by calculating the hypervolumes for each hyperrectangle.

The proposed method is presented in three distinct tasks to address implementation considerations. The hypergrid definition can be either explicitly specified or determined through automated configuration parameters. Counting observations within each hyperrectangle of the chosen hy-

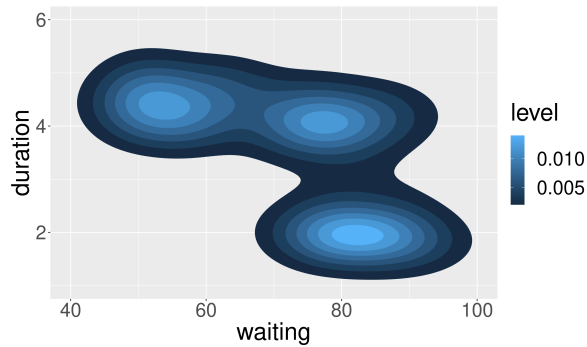


Figure A.6: Density surface

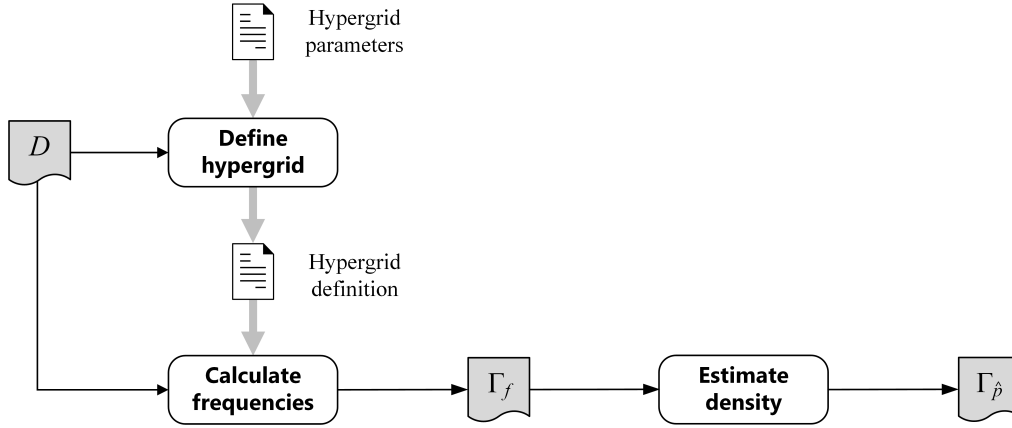


Figure A.7: Overview of the multivariate histogram-based approach

pergrid constitutes the second stage, a relatively straightforward computational task. However, calculating the hypervolumes of these hyperrectangles is computationally more intensive and is only performed for those with a relative frequency greater than zero.

A.2.1 Empirical domains

A dataset, formally defined as a collection of n examples (Equation 2.4), comprises individual examples (e_i), each holding values for a set of d variables (Equation 2.1). For a specific variable x_j within a dataset, O_j is the set encompassing all observed values for that variable across the n examples (Equation A.11).

$$O_j = \{v_{1,j}, \dots, v_{n,j}\} \quad (\text{A.11})$$

The *empirical domain* of a variable, denoted by $\hat{\text{dom}}x_j$, refers specifically to the distinct values observed for that variable within a particular dataset (Holmén, 2020). In other words, it defines the range of values the variable has assumed within that specific data collection. The method to determine the empirical domain depends on the type of the variable:

- Discrete empirical domains – For categorical variables, the empirical domain is defined by the set of unique values observed across all examples (Equation A.12). Consequently, it only encompasses the distinct categories that were actually included in the dataset.
- Continuous empirical domains – For numerical variables, which can be either integer or continuous, the empirical domain is defined as a closed interval (Equation A.13). This interval is bounded by the minimum and maximum values observed for that variable within the dataset.

$$\hat{\text{dom}}x_j = \text{unique}(O_j) \quad (\text{A.12}) \quad \hat{\text{dom}}x = [\min(O_j); \max(O_j)] \quad (\text{A.13})$$

The process of creating bins involves transforming data into a set of discrete sets. These can be defined differently depending on the variable's underlying empirical domain:

- Categorical bins – For categorical variables, a bin represents a set containing m distinct values (Equation A.14). These values correspond to the unique categories the variable can take on within the data.
- Numerical bins – For numerical variables, a bin defines a half-open interval bounded by two specific limits, a and b (Equation A.15). This interval represents a particular range of values on the numerical scale.

$$b_{i,k} = \{c_1, c_2, \dots, c_m\} \quad (\text{A.14})$$

$$b_{i,k} = (a; b] \quad (\text{A.15})$$

A.2.2 Hypergrid

A *hyperrectangle*, denoted as h_k , is a d -dimensional subspace defined as a combination (represented as a tuple) of individual bins, one for each dimension (Equation A.16).

$$h_k = (b_{1,k}, b_{2,k}, \dots, b_{d,k}) \quad (\text{A.16})$$

Partitioning the entire multidimensional space (hyperspace) into a collection of non-overlapping hyperrectangles creates a structure referred to as a *hypergrid*. This process essentially divides the data space into distinct regions. Each region is defined by the specific combinations of binned values for each variable within the data.

Two properties of a hypergrid relevant to its application in multivariate histograms include:

- Non-overlapping hyperrectangles – Adjacent hyperrectangles within the hypergrid never share common boundaries. This ensures that each example falls exclusively within a single hyperrectangle, guaranteeing mutual exclusivity.
- Exhaustiveness – The hypergrid encompasses the entire multidimensional space defined by the empirical domains of all variables in the dataset. This ensures that every example is always contained within a hyperrectangle in the hypergrid.

The creation of a hypergrid can be automated by specifying *hypergrid parameters* for each variable. In the case of categorical variables, each bin corresponds to a distinct value within the empirical domain. For numerical variables, the only mandatory parameter is the number of bins, which are configured to have roughly the same relative frequency. Additionally, for continuous variables, precision (number of decimal places) must be specified (setting it to zero results in discretization into an integer). The outcome is a *hypergrid definition*, specifying, for each variable, the type and corresponding breakpoints (the limits for the bins). Although the hypergrid is initially generated automatically, manual adjustments can be made to fine-tune the breakpoints.

Table A.1 illustrates these concepts for a dataset with three variables: Age (x_1), Gender (x_2), and Income (x_3). For the numerical variables (x_1 and x_3), the creation of bins was automated to create three bins (with the x_3 discretized). For the categorical variable (x_2), a bin was created for

Table A.1: Hypergrid configuration

Variable	Type	Empirical domain	Precision
$x_1 = \text{Age}$	Numerical discrete	[18; 80]	0
$x_2 = \text{Gender}$	Categorical	{Female, Male}	
$x_3 = \text{Income}$	Numerical continuous	[700.5; ...; 2340.7]	

Table A.2: Hypergrid definition

Variable	b_1	b_2	b_3
$x_1 = \text{Age}$	(17; 35]	(35; 52]	(52; 80]
$x_2 = \text{Gender}$	{Female}	{Male}	
$x_3 = \text{Income}$	(699; 1150]	(1150; 1495]	(1495; 2341]

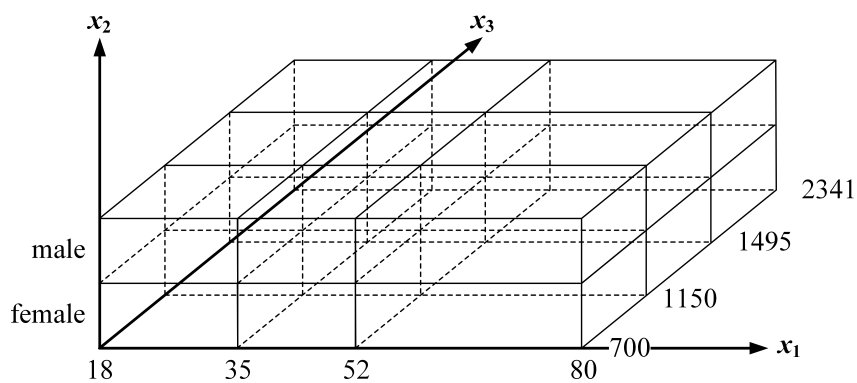


Figure A.8: A 3-D hypergrid with 18 hyperrectangles

each value. The partitioning resulted in the hypergrid definition of Table A.2. Figure A.8 depicts the resulting hypergrid, partitioned into 18 distinct hyperrectangles:

$$\begin{aligned}
 h_1 &= (b_{1,1}, b_{2,1}, b_{3,1}) = ((17; 35], \{\text{Female}\}, (699; 1150]) \\
 h_2 &= (b_{1,1}, b_{2,1}, b_{3,2}) = ((17; 35], \{\text{Female}\}, (1150; 1495]) \\
 h_3 &= (b_{1,1}, b_{2,1}, b_{3,3}) = ((17; 35], \{\text{Female}\}, (1495; 2341]) \\
 h_4 &= (b_{1,1}, b_{2,2}, b_{3,1}) = ((17; 35], \{\text{Male}\}, (699; 1150]) \\
 h_5 &= (b_{1,1}, b_{2,2}, b_{3,2}) = ((17; 35], \{\text{Male}\}, (1150; 1495]) \\
 h_6 &= (b_{1,1}, b_{2,2}, b_{3,3}) = ((17; 35], \{\text{Male}\}, (1495; 2341]) \\
 h_7 &= (b_{1,2}, b_{2,1}, b_{3,1}) = ((35; 52], \{\text{Female}\}, (699; 1150]) \\
 h_8 &= (b_{1,2}, b_{2,1}, b_{3,2}) = ((35; 52], \{\text{Female}\}, (1150; 1495]) \\
 h_9 &= (b_{1,2}, b_{2,1}, b_{3,3}) = ((35; 52], \{\text{Female}\}, (1495; 2341]) \\
 h_{10} &= (b_{1,2}, b_{2,2}, b_{3,1}) = ((35; 52], \{\text{Male}\}, (699; 1150]) \\
 h_{11} &= (b_{1,2}, b_{2,2}, b_{3,2}) = ((35; 52], \{\text{Male}\}, (1150; 1495]) \\
 h_{12} &= (b_{1,2}, b_{2,2}, b_{3,3}) = ((35; 52], \{\text{Male}\}, (1495; 2341]) \\
 h_{13} &= (b_{1,3}, b_{2,1}, b_{3,1}) = ((52; 80], \{\text{Female}\}, (699; 1150]) \\
 h_{14} &= (b_{1,3}, b_{2,1}, b_{3,2}) = ((52; 80], \{\text{Female}\}, (1150; 1495]) \\
 h_{15} &= (b_{1,3}, b_{2,1}, b_{3,3}) = ((52; 80], \{\text{Female}\}, (1495; 2341]) \\
 h_{16} &= (b_{1,3}, b_{2,2}, b_{3,1}) = ((52; 80], \{\text{Male}\}, (699; 1150]) \\
 h_{17} &= (b_{1,3}, b_{2,2}, b_{3,2}) = ((52; 80], \{\text{Male}\}, (1150; 1495]) \\
 h_{18} &= (b_{1,3}, b_{2,2}, b_{3,3}) = ((52; 80], \{\text{Male}\}, (1495; 2341])
 \end{aligned}$$

A.2.3 Multivariate density estimator

The *frequency* (v_k) is the number of examples (e_i) from a dataset D enclosed by a hyperrectangle h_k (Equation A.17). The *relative frequency* (f_k) is the proportion of those in relation to the total number of examples (n) in the dataset (Equation A.18).

$$v_k = |\{e_i : e_i \in D \wedge e_i \in h_k\}| \quad (\text{A.17}) \quad f_k = \frac{v_k}{n} \quad (\text{A.18})$$

A *multivariate histogram*, denoted by $\Gamma_{\hat{p}}$, results from the K pairs of hyperrectangles and corresponding relative frequencies (Equation A.19).

$$\Gamma_f = \{(h_k, f_k), k \in [1; K]\} \quad (\text{A.19})$$

Density estimation is performed directly from the previously calculated multivariate histogram by calculating the hypervolume of each hyperrectangle. The *hypervolume* (V_k) is a measure of the volume of a hyperrectangle, a generalization of the *bin width* of 1-dimensional histograms (Luo

and Mehrotra, 2018). Formally, a hypervolume is the product of the width ($w_{j,k}$) of the d edges of a hyperrectangle h_k (Equation A.22). For numerical variables (either discrete or continuous), the width is the difference between the upper and lower bounds of the edge (Equation A.20). For categorical variables, the width is the number of distinct values in the edge (Equation A.21).

$$w_{j,k} = \max b_{j,k} - \min b_{j,k} \quad (\text{A.20}) \quad w_{j,k} = |b_{j,k}| \quad (\text{A.21})$$

The *density estimate* ($\hat{p}_k$) is the average number of examples in a unit of hypervolume. Formally, it is the relative frequency normalized by the hypervolume (Equation A.23). Consequently, unlike the cumulative sum of relative frequencies, the density estimates of all the hyperrectangles do not add up to one.

$$V_k = \prod_{j=1}^d w_{j,k} \quad (\text{A.22}) \quad \hat{p}_k = \frac{f_k}{V_k} \quad (\text{A.23})$$

A *multivariate density estimator*, denoted by $\Gamma_{\hat{p}}$, results from the K pairs of hyperrectangles and corresponding density estimates (Equation A.24).

$$\Gamma_{\hat{p}} = \{(h_k, \hat{p}_k), k \in [1; K]\} \quad (\text{A.24})$$

Table A.3 presents a density estimator based on the hypergrid defined in Table A.1. For brevity, only hyperrectangles with non-zero density estimates ($\hat{p}$) are included. Table A.4 lists all the intermediate calculations used to derive the density estimates, namely, frequencies (v), relative frequencies (f), widths of hyperrectangle edges (w), and hypervolumes (V).

Table A.3: A density estimator based on a multivariate histogram ($\Gamma_{\hat{p}}$)

k	b_1	b_2	b_3	$\hat{p}$
1	(17;35]	{Female}	(699;1150]	1.6×10^{-5}
4	(17;35]	{Male}	(699;1150]	1.2×10^{-5}
8	(35;52]	{Female}	(1150;1495]	5.0×10^{-5}
9	(35;52]	{Female}	(1495;2341]	0.3×10^{-5}
11	(35;52]	{Male}	(1150;1495]	3.6×10^{-5}
14	(52;80]	{Female}	(1150;1495]	0.6×10^{-5}
15	(52;80]	{Female}	(1495;2341]	0.2×10^{-5}
17	(52;80]	{Male}	(1150;1495]	0.8×10^{-5}
18	(52;80]	{Male}	(1495;2341]	0.2×10^{-5}

A.3 Handling datasets with missing data

A feature of the proposed method is its ability to estimate density even in datasets with missing data. If a dataset has missing data, the task to calculate frequencies is extended with two subtasks, depicted in Figure A.9, in which there are three versions of the multivariate histogram. The first is

Table A.4: Density estimator intermediate calculations

k	ν	f	w_1	w_2	w_3	V
1	235	0.126	18	1	451	8118
4	183	0.098	18	1	451	8118
8	542	0.291	17	1	345	5865
9	87	0.047	17	1	846	14382
11	390	0.210	17	1	345	5865
14	116	0.062	28	1	345	9660
15	93	0.050	28	1	846	23688
17	146	0.079	28	1	345	9660
18	68	0.037	28	1	846	23688

an *incomplete multivariate histogram* (denoted as $\Gamma_f^{(1)}$), i.e., containing hyperrectangles with edges unknown. A hyperrectangle having one or more unknown edges is an *undefined hyperrectangle*.

The approach for handling missing values is to replace all undefined hyperrectangles with complete ones (without unknown edges), provided that they cover the same region of the subspace. Thus, for each undefined hyperrectangle, it is defined as a *missing values hypergrid*, containing only the variables in the unknown edges.

Figure A.10 shows an example of an undefined hyperrectangle, enclosing all examples where $x_1 \in (52; 80]$ and with missing values in both x_2 and x_3 . A missing values hypergrid is defined from the combination of the bins of x_2 and x_3 (Figure A.11). Then, a *missing values multivariate histogram* (Γ_ϕ) is obtained using an extract of the dataset with examples without missing values on both x_2 and x_3 .

The undefined hyperrectangle is then replaced by a set of *complete hyperrectangles* (Figure A.12). When replaced, the relative frequencies of the complete hyperrectangles are scaled by the relative frequency of the undefined hyperrectangle (Equation A.25).

$$f_{k_2} = \begin{cases} f_{k_1} f_{k_\phi}, & \text{if } k_1 \text{ is undefined} \\ f_{k_1}, & \text{otherwise} \end{cases} \quad (\text{A.25})$$

The result is a *completed histogram* (denoted as $\Gamma_f^{(2)}$). Replacing all undefined hyperrectangles

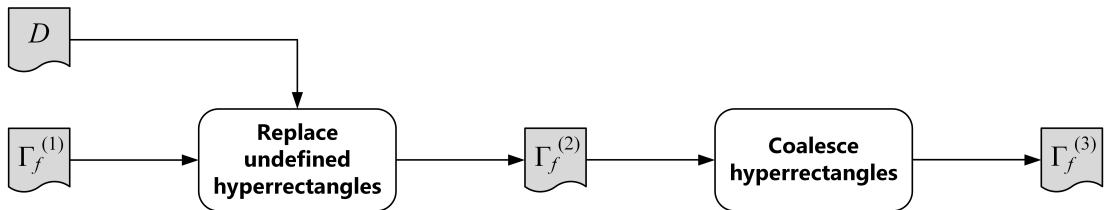


Figure A.9: Subtasks to handle datasets with missing data

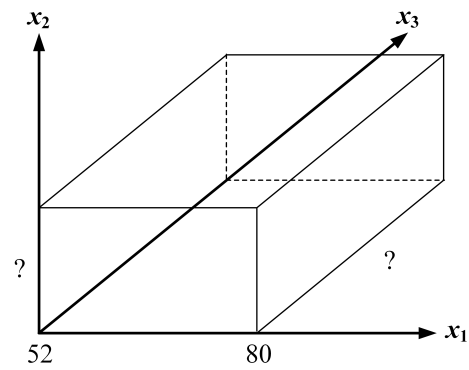


Figure A.10: Undefined hyperrectangle

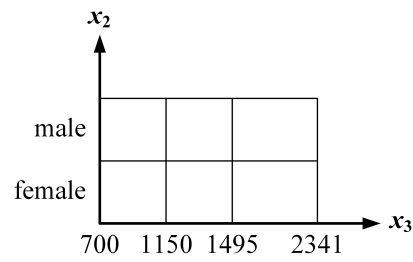


Figure A.11: Missing values 2-D hypergrid

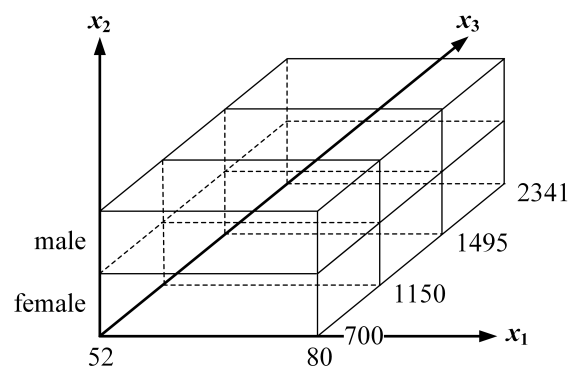


Figure A.12: Complete hyperrectangles

with complete ones can result in multiple instances of the same hyperrectangles. Therefore, it is necessary to coalesce identical hyperrectangles into one by adding their relative frequencies. The result is a *coalesced histogram* (denoted $\Gamma_f^{(3)}$), i.e., without repeated hyperrectangles. The process is then resumed with the task of density estimation.

Table A.5 illustrates the process with a incomplete multivariate histogram ($\Gamma_f^{(1)}$), with an undefined hyperrectangle of Figure A.10 (with index denoted as $k_1 = *1$). Table A.6 presents the missing values histogram (Γ_ϕ), obtained from the examples in which x_2 and x_3 are not missing (in the context of the hypergrid depicted in Figure A.11). The known edges of the undefined hyperrectangle are then combined with the hyperrectangles of Γ_ϕ , resulting in the completed hyperrectangles depicted in Figure A.12. These replace the undefined hyperrectangle with the relative frequencies scaled by the relative frequency of the undefined hyperrectangle, yielding the completed multivariate histogram ($\Gamma_f^{(2)}$) in Table A.7. The relative frequencies of the other hyperrectangles remain unchanged. The replacement of the undefined hyperrectangle led to the addition of new hyperrectangles in $\Gamma_f^{(2)}$ ($k_2 = 13$ and $k_2 = 16$), while the remaining ones cover subsets already in $\Gamma_f^{(1)}$. Hence, the duplicate hyperrectangles are coalesced into one with the relative frequencies added, resulting in the coalesced multivariate histogram ($\Gamma_f^{(3)}$) in Table A.8.

 Table A.5: Incomplete multivariate histogram ($\Gamma_f^{(1)}$)

k_1	b_1	b_2	b_3	v	f
1	(17; 35]	{Female}	(699; 1150]	235	0.113
4	(17; 35]	{Male}	(699; 1150]	183	0.088
8	(35; 52]	{Female}	(1150; 1495]	542	0.260
9	(35; 52]	{Female}	(1495; 2341]	87	0.042
11	(35; 52]	{Male}	(1150; 1495]	390	0.187
14	(52; 80]	{Female}	(1150; 1495]	116	0.056
15	(52; 80]	{Female}	(1495; 2341]	93	0.045
17	(52; 80]	{Male}	(1150; 1495]	146	0.070
18	(52; 80]	{Male}	(1495; 2341]	68	0.032
*1	(52; 80]			223	0.107

A.4 Evaluation on selected datasets

The evaluation of the density estimation method encompasses two perspectives: computational performance and sensitivity to dataset changes. Computational performance involves assessing the capability to estimate density in a number of datasets with distinct characteristics and identifying potential relationships between dataset size and running time. Additionally, given the method's capability to handle missing values and categorical variables, the results of experiments on the impact of these factors on density estimates are presented.

Table A.6: Missing values multivariate histogram (Γ_ϕ)

k_ϕ	b_2	b_3	v	f
1	{Female}	(699; 1150]	235	0.126
2	{Female}	(1150; 1495]	658	0.354
3	{Female}	(1495; 2341]	180	0.097
4	{Male}	(699; 1150]	183	0.098
5	{Male}	(1150; 1495]	536	0.288
6	{Male}	(1495; 2341]	68	0.037

A.4.1 Evaluation of computational performance in density estimation

The method's computational performance was assessed by evaluating density estimation across datasets from the UCI (Markelle Kelly et al., 2023), Kaggle (Kaggle, 2024), and OpenML (Vanschoren et al., 2014) repositories. Table A.9 presenters the selected datasets, described by a range of characteristics, as follows:

- Size (n) – The number of examples;
- Dimensionality (d) – The number of variables (dimensions);
- Volume (μ) – The product of size and dimensionality to capture the amount of data within the dataset (Equation 2.6);
- Categorical variables (d_c) – The number of categorical variables ($d_c \leq d$);
- Size of missing values (n_{mv}) – The number of examples with missing data ($n_{mv} \leq n$);
- Ratio of missing values (r_{mv}) – The proportion of examples with missing data (Equation A.26);

$$r_{mv} = \frac{n_{mv}}{n} \quad (\text{A.26})$$

- Computational time (t) – Running time (in seconds) required for density estimation.

To explore the impact of dimensionality and handle various data types, datasets with increasing dimensionality, categorical variables (present or absent), and missing values (varying proportions) were selected. These considerations led to the categorization of datasets into four groups:

- Group #1 – Low dimensionality ($d \leq 6$), numerical variables only ($d_c = 0$), no missing values ($r_{mv} = 0$);
- Group #2 – High dimensionality ($d > 6$), numerical variables only ($d_c = 0$), no missing values ($r_{mv} = 0$);
- Group #3 – Any dimensions, categorical variables included ($d_c > 0$), no missing values ($r_{mv} = 0$);

Table A.7: Completed multivariate histogram ($\Gamma_f^{(2)}$)

k_2	k_1	k_ϕ	b_1	b_2	b_3	f
1	1		(17;35]	{Female}	(699;1150]	0.113
4	4		(17;35]	{Male}	(699;1150]	0.088
8	8		(35;52]	{Female}	(1150;1495]	0.260
9	9		(35;52]	{Female}	(1495;2341]	0.042
11	11		(35;52]	{Male}	(1150;1495]	0.187
13	*1	1	(52;80]	{Female}	(699;1150]	0.014
14	14		(52;80]	{Female}	(1150;1495]	0.056
14	*1	2	(52;80]	{Female}	(1150;1495]	0.038
15	15		(52;80]	{Female}	(1495;2341]	0.045
15	*1	3	(52;80]	{Female}	(1495;2341]	0.010
16	*1	4	(52;80]	{Male}	(699;1150]	0.011
17	17		(52;80]	{Male}	(1150;1495]	0.070
17	*1	5	(52;80]	{Male}	(1150;1495]	0.031
18	18		(52;80]	{Male}	(1495;2341]	0.032
18	*1	6	(52;80]	{Male}	(1495;2341]	0.004

Table A.8: Coalesced multivariate histogram ($\Gamma_f^{(3)}$)

k_3	b_1	b_2	b_3	f
1	(17;35]	{Female}	(699;1150]	0.113
4	(17;35]	{Male}	(699;1150]	0.088
8	(35;52]	{Female}	(1150;1495]	0.260
9	(35;52]	{Female}	(1495;2341]	0.042
11	(35;52]	{Male}	(1150;1495]	0.187
13	(52;80]	{Female}	(699;1150]	0.014
14	(52;80]	{Female}	(1150;1495]	0.094
15	(52;80]	{Female}	(1495;2341]	0.055
16	(52;80]	{Male}	(699;1150]	0.011
17	(52;80]	{Male}	(1150;1495]	0.101
18	(52;80]	{Male}	(1495;2341]	0.036

Table A.9: Datasets selected for the density estimation method evaluation

Gr.	Dataset	n	d	μ	d_c	n_{mv}	r_{mv}	t
#1	China’s population★	73	4	292	0	0	0.00	0.04
	Iris†	150	4	600	0	0	0.00	0.05
	Old faithful★	242	2	484	0	0	0.00	0.06
	Phoneme◇	5 404	5	27 020	0	0	0.00	0.16
	2D elastod. metamat.†	20 520	3	61 560	0	0	0.00	0.37
	Stock market@Kraken★	32 946 819	2	65 893 638	0	0	0.00	264.89
#2	Wine quality★	1 143	12	13 716	0	0	0.00	0.28
	Dry bean†	13 611	16	217 776	0	0	0.00	1.16
	SGEMM GPU kernel perf.†	241 600	18	4 434 800	0	0	0.00	45.85
	Students’ dropout†	4 424	37	163 688	0	0	0.00	1.62
	Rice MSC★	75 000	106	7 950 000	0	8	0.00	214.05
	Biological response†	3 751	1776	6 661 776	0	0	0.00	78.81
#3	Height of male/Female★	199	4	796	1	0	0.00	0.14
	Students perf. in exams★	1 000	8	8 000	5	0	0.00	0.20
	Car evaluation†	1 728	7	12 096	7	0	0.00	0.42
	Auction verification†	2 043	9	18 387	1	0	0.00	0.25
	Diabetes†	29 278	2	58 556	1	0	0.00	0.19
	Airlines train◇	10 000 000	10	100 000 000	3	0	0.00	1162.48
#4	Breast cancer Wisconsin†	699	8	5 592	0	16	0.02	0.14
	HCV◇	615	13	7 995	2	26	0.04	0.79
	Astronauts★	952	11	10 472	10	952	1.00	13.77
	Mushroom†	8 124	23	186 852	23	2 480	0.31	4.71
	Job change of data scient.★	19 158	12	229 896	10	10 203	0.53	262.36
	Adult†	32 561	13	423 293	8	2 399	0.07	31.23

★ Kaggle (Kaggle, 2024); † UCI (Markelle Kelly et al., 2023); ◇ OpenML (Vanschoren et al., 2014)

- Group #4 – Any dimensions, any variable types, missing values present ($r_{mv} > 0$).

In all datasets, the underlying hypergrid was created to contain three bins in numerical variables and one separate bin for each value in categorical variables. The results are hardly surprising. The most penalizing factors in density estimation computational time are the dataset size and the ratio of missing values. Nevertheless, the analysis reveals distinctions between datasets based on their size. Datasets are categorized as small when $\mu < 30000$, and large otherwise. Although the selection of datasets is not very broad, these results indicate that the calculation effort tends to be linear to the size of the dataset.

Figure A.13 illustrates this trend in smaller datasets, with most groups exhibiting nearly flat regression lines. The exception is group #4 (datasets with missing values), where the slope is significantly higher. This difference arises from the additional computational load of handling missing data, which often requires extensive hyperrectangle replacements. For instance, the As-

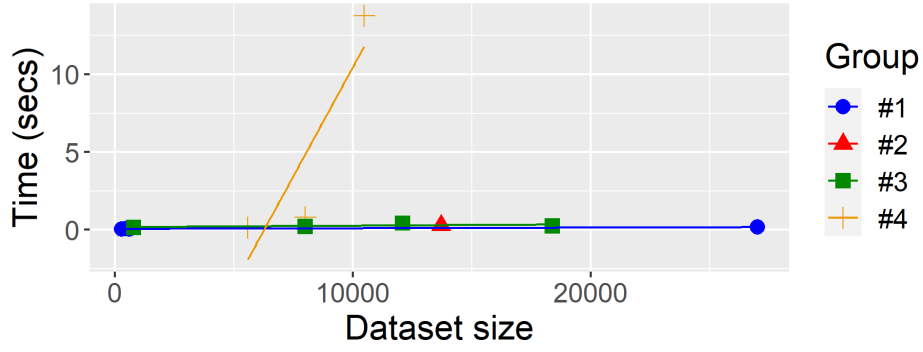


Figure A.13: Smaller datasets density estimation computational time ($\mu < 30000$)

tronauts dataset, where all observations have at least one missing value, demonstrates a significant increase in computational time (approximately 13 seconds).

In contrast, larger datasets (Figure A.14) show a clear and consistent linear relationship between dataset size and density estimation computational time. This trend holds even for group #4, indicating that the impact of missing values becomes less significant as the overall dataset size increases.

A.4.2 Evaluation of the impact of categorical variables in density estimation

The next set of experiments investigates the influence of introducing categorical variables on the density estimates within hyperrectangles. Two effects are expected:

- Data space partitioning – Introducing a new variable, especially a categorical one, is expected to distribute examples across more hyperrectangles. This reflects the finer partitioning of the data space based on the added information.
- Density decrease – As the dimensionality of the data increases, the density estimates associated with individual hyperrectangles are expected to exhibit a general decrease. This is a consequence of the data becoming increasingly sparse in a higher dimensional space.

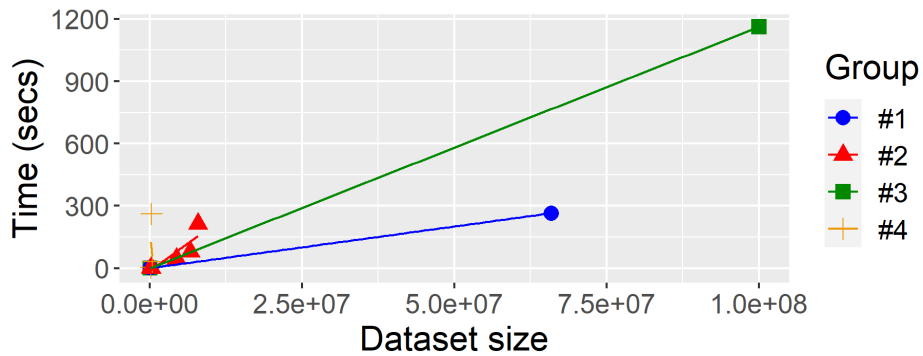


Figure A.14: Larger datasets density estimation computational time ($\mu \geq 30000$)

Table A.10: Density estimator without categorical variables ($d_c = 0$)

k	b_1	b_2	$\hat{p}$
1	(1.5; 2.4]	(42; 64]	0.01904
2	(1.5; 2.4]	(64; 80]	0.00115
4	(2.4; 4.3]	(42; 64]	0.00064
5	(2.4; 4.3]	(64; 80]	0.00617
6	(2.4; 4.3]	(80; 96]	0.00435
8	(4.3; 5.1]	(64; 80]	0.01120
9	(4.3; 5.1]	(80; 96]	0.01379

As the number of examples remains unchanged, the sum of densities within the newly formed hyperrectangles is expected to necessarily match the density of the larger, encompassing hyperrectangles that held those examples prior to the variable’s introduction. This principle guarantees that the overall representation of the data remains consistent despite the refined density estimates within the new hyperrectangles.

The analysis used the Old Faithful dataset (Kaggle, 2024) (Section A.1.3) to control the effects of adding new variables on density estimation. Categorical variables were then successively introduced. Table A.10 illustrates a bivariate density estimator with three bins for numerical variables x_1 and x_2 , where only hyperrectangles with non-zero density estimates ($\hat{p} > 0$) are shown.

In the first experiment, a categorical variable (x_3) was added. This variable was randomly assigned two values (c1 and c2), resulting in the density estimator presented in Table A.11. Examples initially within each hyperrectangle from Table A.10 are now distributed across two separate hyperrectangles in Table A.11, one for each value of x_3 . Additionally, the sum of the density estimates for each corresponding pair of hyperrectangles in Table A.11 (sharing the same bins for x_1 and x_2 but differing in the bin for x_3) equals the density estimate of the original, aggregating hyperrectangle in Table A.10. For instance, $\hat{p}_1$ in Table A.10 equals $\hat{p}_1 + \hat{p}_2$ in Table A.11. This ensures that the overall data mass is conserved despite the changes in density estimates within individual hyperrectangles due to the introduction of the new categorical variable.

Subsequent experiments introduced a second and then a third categorical variable, x_4 and x_5 , respectively. Both variables were randomly assigned the same values. The resulting density estimators are presented in Tables A.12 and A.13. As before, the sum of the density estimates for each corresponding pair of hyperrectangles in these tables (sharing the same bins for previous numerical variables but differing in the bin for the newly added variable) must equal the density estimate of the original, aggregating hyperrectangle from the previous dimensionality (e.g., Table A.11).

This controlled introduction of binary categorical variables allows for the analysis of the impact of increasing dimensionality on density estimation within the hypergrid framework. As new categorical variables are added, the hypergrid becomes denser, with more hyperrectangles partitioning the data space. Consequently, the average density within each hyperrectangle, and across the entire data space, tends to decrease.

Table A.11: Density estimator with one categorical variable ($d_c = 1$)

k	b_1	b_2	b_3	$\hat{p}$
1	(1.5;2.4]	(42;64]	{c1}	0.00810
2	(1.5;2.4]	(42;64]	{c2}	0.01094
3	(1.5;2.4]	(64;80]	{c1}	0.00029
4	(1.5;2.4]	(64;80]	{c2}	0.00086
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
17	(4.3;5.1]	(80;96]	{c1}	0.00603
18	(4.3;5.1]	(80;96]	{c2}	0.00776

Table A.12: Density estimator with two categorical variables ($d_c = 2$)

k	b_1	b_2	b_3	b_4	$\hat{p}$
1	(1.5;2.4]	(42;64]	{c1}	{c1}	0.00219
2	(1.5;2.4]	(42;64]	{c1}	{c2}	0.00591
3	(1.5;2.4]	(42;64]	{c2}	{c1}	0.00416
4	(1.5;2.4]	(42;64]	{c2}	{c2}	0.00678
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
35	(4.3;5.1]	(80;96]	{c2}	{c1}	0.00287
36	(4.3;5.1]	(80;96]	{c2}	{c2}	0.00488

Table A.13: Density estimator with three categorical variables ($d_c = 3$)

k	b_1	b_2	b_3	b_4	b_5	$\hat{p}$
1	(1.5;2.4]	(42;64]	{c1}	{c1}	{c1}	0.00131
2	(1.5;2.4]	(42;64]	{c1}	{c1}	{c2}	0.00088
3	(1.5;2.4]	(42;64]	{c1}	{c2}	{c1}	0.00219
4	(1.5;2.4]	(42;64]	{c1}	{c2}	{c2}	0.00372
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
71	(4.3;5.1]	(80;96]	{c2}	{c2}	{c1}	0.00201
72	(4.3;5.1]	(80;96]	{c2}	{c2}	{c2}	0.00287

A.4.3 Evaluation of the impact of missing data in density estimation

To investigate the influence of missing values on the density estimates within hyperrectangles, a set of experiments creates new hyperrectangles to represent examples with missing data, even in subspaces where there were no examples. As the proportion of missing data increases, the hypothesis is that the density of these newly created hyperrectangles will rise correspondingly, potentially accompanied by variations in the densities of existing ones.

As previously, The Old Faithful dataset (Kaggle, 2024) was used again as test data. Values in the original dataset were progressively replaced with missing values to control the ratio of missing data (r_{mv}). Table A.14 presents the results for three ratios of missing data: $r_{mv} = 0.01$ ($\hat{p}_1$), $r_{mv} = 0.05$ ($\hat{p}_2$), and $r_{mv} = 0.10$ ($\hat{p}_3$). The table also shows the difference in density ($\Delta\hat{p}$) compared to the original dataset with no missing values (Table A.10).

With a ratio of missing data of 1% ($r_{mv} = 0.01$), the differences in density estimates are relatively small, typically on the order of 10^{-4} or less (including the newly created hyperrectangles #3 and #7, which originally had zero density and were omitted from Table A.10). This minimal impact is likely due to the low number of examples (three) with missing values added. However, for higher ratios of missing data ($r_{mv} = 0.05$ and $r_{mv} = 0.10$), the variations in density, while still within the same order of magnitude, become more pronounced.

These results support the hypothesis that the contribution of hyperrectangles derived from missing values becomes progressively more significant in the overall density estimates as the proportion of missing values increases. As more examples with missing data are introduced, the corresponding hyperrectangles become denser, potentially affecting the densities of previously occupied regions due to the redistribution of data points within the hypergrid.

Table A.14: Density estimator with controlled addition of missing data ($r_{mv} > 0$)

k	b_1	b_2	$\hat{p}_1$	$\Delta\hat{p}_1$	$\hat{p}_2$	$\Delta\hat{p}_2$	$\hat{p}_3$	$\Delta\hat{p}_3$
1	(1.5;2.4]	(42;64]	0.01890	-1.4×10^{-4}	0.01817	-8.7×10^{-4}	0.01796	-10.8×10^{-4}
2	(1.5;2.4]	(64;80]	0.00135	2.0×10^{-4}	0.00154	3.9×10^{-4}	0.00174	5.9×10^{-4}
3	(1.5;2.4]	(80;96]	0.00009	0.9×10^{-4}	0.00055	5.5×10^{-4}	0.00103	1.0×10^{-4}
4	(2.4;4.3]	(42;64]	0.00064	0.0×10^{-4}	0.00077	1.3×10^{-4}	0.00084	2.0×10^{-4}
5	(2.4;4.3]	(64;80]	0.00609	-0.8×10^{-4}	0.00613	-4.0×10^{-4}	0.00605	-1.2×10^{-4}
6	(2.4;4.3]	(80;96]	0.00435	0.0×10^{-4}	0.00403	-3.2×10^{-4}	0.00394	-4.1×10^{-4}
7	(4.3;5.1]	(42;64]	0.00008	0.8×10^{-4}	0.00044	4.4×10^{-4}	0.00081	8.1×10^{-4}
8	(4.3;5.1]	(64;80]	0.01139	1.9×10^{-4}	0.01102	-1.8×10^{-4}	0.01083	-3.7×10^{-4}
9	(4.3;5.1]	(80;96]	0.01359	-2.0×10^{-4}	0.01405	2.6×10^{-4}	0.01353	-2.6×10^{-4}

A.5 Summary

Density estimation methods have several applications in the scope of exploratory data analysis. Non-parametric approaches are limited to a small number of dimensions. Histogram-based ap-

proaches, despite less accurate density estimates, remain the only alternative concerning datasets in high-dimensional spaces. However, current implementations still have constraints based on the dimensionality of a dataset.

The work presented in this appendix is a novel approach that overcomes these limitations by enabling density estimation on datasets with no restriction on the number of variables. It effectively handles both categorical variables and missing data, with the only constraint being the overall size of the dataset (product of observations and dimensions). The empirical evaluation not only demonstrates the method's ability to handle high-dimensional data but also corroborates its expected behavior in the presence of categorical variables and missing values.

Appendix B

Measuring Completeness: An Evaluation for Rule-Based Models

In machine learning, the term *completeness* encompasses various meanings, including dataset quality, rule coverage, black-box model explainability, and overall performance. This appendix focuses on completeness as a verifiable characteristic of rule-based models (Ligeza, 2006). In a nutshell, a model is considered complete if it consistently provides predictions for all possible inputs within its decision space. Incompleteness manifests as uncovered examples, which are data points for which the model cannot make a prediction, often suggesting missing rules.

Chapter 4 presents the development of consensus models by merging rule-based models. However, the merging process introduces challenges related to completeness, potentially resulting in incomplete coverage. This limitation can impact the model’s practical value for real-world decision-making within an organization. This appendix introduces two novel metrics for a more rigorous evaluation of completeness, specifically within the *Inmplode* framework for consensus model development. The first metric examines geometrical completeness of the decision space, while the second evaluates completeness relative to data concentration, using the density estimation method detailed in Appendix A. Both metrics are evaluated on selected datasets during the model merging process.

B.1 Completeness in machine learning

Some of the most common interpretations of completeness involve broader categories such as data completeness, completeness in rule’s coverage, model completeness, completeness of explanations, and completeness as a performance measure, described next.

B.1.1 Completeness in data

Data completeness encompasses two distinct problems: the first is the occurrence of missing data where it was expected, and the second is the inadequacy of datasets sufficiently representing the decision space.

Missing data is a significant data quality issue, alongside accuracy and consistency, and occurs across various domains (Han et al., 2012). It can result from human errors during data processing, equipment malfunctions, non-responses in surveys, dropout rates, and privacy concerns (Houari et al., 2014; Emmanuel et al., 2021). Missing values often lead to biased outcomes, reduced performance, and challenges in data analysis, with their impact influenced by the extent, patterns, and underlying mechanisms of the missing data (Ayilara et al., 2019; Kang, 2013). Addressing these challenges is necessary to minimize disparities between complete and incomplete datasets.

Techniques for managing missing data include deleting incomplete examples or using *imputation*, where missing values are estimated (Langkamp et al., 2010; Donders et al., 2006). Common approaches include mean imputation, maximum likelihood estimation, regression, K-nearest neighbor, and ensemble-based methods (Graham, 2009; Baraldi and Enders, 2010; Kang, 2013; Fürnkranz et al., 2012). Hybrid methods have been proposed to mitigate the limitations of traditional techniques (Aydilek and Arslan, 2013; Al-Helali et al., 2021). In the era of big data, traditional methods often face scalability issues due to their design for smaller datasets. Thoughtful algorithm design and careful attribute and sampling selection are essential to ensure reliable and accurate outcomes (Qiu et al., 2016; Emmanuel et al., 2021).

The second aspect of data completeness concerns the adequacy of a dataset in accurately representing a given phenomenon, often linked to selection bias (Bareinboim and Tian, 2015). Selection bias arises when nonrandom sampling results in data that are not statistically representative of the population (Collier, 1995). A dataset is incomplete when its observations fail to approximate the distribution of the underlying population, limiting the decision space and hindering models, such as rule-based systems, from addressing the full range of real-world scenarios. This bias persists even in large datasets if rare events are not adequately captured (Wulff et al., 2018).

Addressing this issue requires ensuring that datasets realistically represent the decision space, often through collaboration with domain experts to mitigate selection bias in observational studies (Smith and Vanderweele, 2019). While missing data and selection bias are distinct problems, they are interrelated. Missing data can contribute to model incompleteness if the available observations do not adequately cover the decision space of the problem under analysis.

B.1.2 Completeness in rule-based models

In the context of rule-based models, completeness has appeared in two distinct settings. The first is in relation to the coverage of rules, establishing a distinction between *complete rules* (which cover more positive examples) and *consistent rules* (which cover fewer negative examples) (Cohen, 1995; Fürnkranz and Flach, 2005). Striking a balance between completeness and consistency is important for these algorithms. While maximizing completeness is desirable, achieving it totally can be counterproductive as it may lead to overfitting. Pruning emerges as a common strategy to address this, introducing a bias towards simpler rules to avoid overly complex ones with low coverage (Fürnkranz, 1997). By incorporating this bias, pruning reduces overfitting, resulting in a more generalizable model, albeit with potentially lower completeness.

Another perspective on completeness in rule-based models views it as one of the properties verifiable within a *Taxonomy of verifiable characteristics* (Ligeza, 2006), which encompasses four characteristics, as follows:

- Redundancy – Detection of identical rules (if rules are exactly the same), subsumed rules (if one rule is more generic than another and can therefore replace it), equivalent rules (if one rule subsumes the other and vice-versa), and unusable rules (if a rule for some reason is never activated).
- Consistency – Detection of indeterminism leading to ambiguous rules (if two or more rules can be applied for the same input, but their predictions are different), conflict leading to ambivalent rules (a particular case of the previous one in which the predictions cannot be both correct from the business point of view).
- Reduction – Detection if rules can be reduced by replacing two or more rules with an equivalent rule, or by elimination of unnecessary variables (either using knowledge about functional dependencies or as a consequence of reduction).
- Completeness – Detection if all possible values of the independent variables satisfy at least one rule. If a model fails this property, it is considered the basis for a missing rule (Suwa et al., 1982). This problem is known as *uncovered examples* when no rule covers a new example (Fürnkranz et al., 2012).

Decision trees, due to the property of exhaustiveness, are always complete models as there is one rule for every combination of variables and values (Han et al., 2012). In other words, all rules collectively cover the entire decision space. In the case of classification rules, exhaustiveness is not guaranteed. Therefore, a common strategy to achieve completeness if a model is incomplete is to include a default rule for regions of the decision space not represented in the training set (Fürnkranz et al., 2012; Alonso Moral, 2007).

B.1.3 Completeness in fuzzy expert systems

Fuzzy Expert Systems (FES) (Zadeh, 1983) stand out from traditional rule-based systems due to their reliance on fuzzy logic. This enables them to handle the inherent uncertainties and ambiguities of real-world issues, making them ideal for tasks like medical diagnosis, where data often spans a spectrum. Unlike classical (or crisp) sets with their binary *included* or *not included* classifications, fuzzy sets introduce graded membership, allowing elements to belong with varying degrees partially (Pedrycz and Gomide, 2007). This flexibility forms the foundation of fuzzy rules, expressed as *If-Then* statements with fuzzy conditions. Derived from human expertise, these rules enable FES to mimic human reasoning and decision-making in complex situations (Tavana and Hajipour, 2019). However, ensuring the reliability and interpretability of these rules relies on completeness and consistency.

Similar to covering algorithms, completeness in FES demands that every possible input scenario be covered by at least one fuzzy rule. Conversely, consistency dictates that an input satisfying a rule would not belong to any other class within the system. As in rule-based models, achieving both remains challenging due to the risk of overfitting. Completeness plays a pivotal role in FES, guaranteeing the system’s ability to handle any conceivable input effectively. This stems from the inherent flexibility of fuzzy sets, which allows FES to encompass the entirety of the potential data domain. Essentially, it guarantees a *prepared response* for any input scenario, even unforeseen ones. The pursuit of achieving complete coverage in FES remains an active area of research (Mendel, 2017).

B.1.4 Completeness in interpretability

Completeness is considered an important aspect of interpretability, ensuring that an explanation covers all factors contributing to a prediction. Recent advancements in this field include interpretable machine learning (Carvalho et al., 2019), argumentation-based explanation models (Vassiliades et al., 2021), and the development of comprehensive evaluation metrics for interpretability and completeness (Farah et al., 2023). Interpretability seeks to make the inner workings of a system understandable to humans, while completeness is considered to yield an accurate depiction of the system’s operation. For instance, concept-based explanations aim to characterize the global behavior of Deep Neural Networks (DNNs) (Yeh et al., 2020), while bipolar abstract argumentation frameworks represent arguments within a graph to depict the impact of variables in a classification problem (Potyka et al., 2022). There are different ways to evaluate completeness, such as comparing it to the original model or assessing it in relation to a substitute task (Gilpin et al., 2018).

Another perspective frames completeness as one of the two important aspects of explanation proposed (Kulesza et al., 2013). The first is soundness, as the extent to which each component of an explanation’s content is truthful in describing the underlying system, or in other words, truthfulness associated with each component of the system (Dutta et al., 2021), more aligned with local interpretability. The second is completeness, as the extent to which the explanation describes all of the underlying systems more aligned with global interpretability. Completeness has also been referred to as one of “the three Cs of interpretability” (Silva et al., 2018), in the context of properties that a good explanation should maximize. Specifically, completeness is indicated as the susceptibility of the explanation that can be verified in any example that can confirm it. The other two are correctness, evaluating if the explanation is correct (akin to fidelity), and compactness, related to the importance of succinct explanations (measured through the length of the rules).

B.1.5 Completeness in performance evaluation

Completeness is sometimes used as a measure of a model’s performance based on error. While Precision can be thought of as a measure of exactness (i.e., what percentage of tuples labeled as positive are actually such), whereas recall is a measure of completeness (what percentage of

positive tuples are labeled as such) (Han et al., 2012). However, in other contexts, completeness emerges as a metric for assessing the effectiveness of a model in reducing predictive error compared to both a simple baseline and an upper bound determined by the inherent complexity of the data.

Generally, the predictive error of a model can be separated into three components: bias, variance, and irreducible noise (Van Der Valk and Picek, 2019). The bias represents the difference between the expected prediction of the model and the actual value to predict. High bias indicates that the model struggles to capture the underlying relationships in the data, resulting in underfitting. On the other hand, variance accounts for the variability in model predictions for a given example. High variance suggests that the algorithm is modeling the noise in the training data and may fail to generalize to new, unseen data, indicative of overfitting. Finally, irreducible noise represents the fundamental limitations of any model based on the available data. This inherent noise arises from factors not captured by the chosen independent variables, setting a lower bound on model error achievable by any algorithm.

When analyzing prediction errors, it is common to focus on bias and variance, neglecting the irreducible noise present in the data. Finding the optimal balance between these two elements is important, and this duality is often referred to as the *bias-variance trade-off* (Brady and Brockmeier, 2018). In other words, an underfitted model is not complex enough to capture important patterns in the data, leading to poor performance. Conversely, a model with excessive complexity can overfit the data, capturing noise instead of learning generalizable patterns, which also leads to poor performance on unseen data.

In this context, completeness measures a model’s performance (ϵ_m) compared to two extremes. The first is irreducible noise (ϵ_i), which represents the inherent limitations of the data and sets the best possible performance for any model using those variables (Fudenberg et al., 2022). The other extreme is the performance of a naive baseline model (ϵ_n), which essentially predicts randomly. Completeness is defined as the proportion of achievable error reduction achieved by the model (Equation B.1). A completeness of 0 indicates the model performs no better than the naive baseline, while a completeness of 1 implies that the model eliminates all but the irreducible noise.

$$C_m = \frac{\epsilon_n - \epsilon_m}{\epsilon_n - \epsilon_i} \quad (\text{B.1})$$

B.2 A taxonomy of decision spaces

A *decision space* (Giabbanelli and Peters, 2015) is a multidimensional space where each dimension corresponds to a distinct attribute or characteristic that influences a decision. Data points within this space represent unique combinations of attribute values and their corresponding outcomes. It is relevant to distinguish between decision space and sample space (Smithson and Shou, 2021). While a decision space focuses on the independent variables influencing a decision, the sample space refers to the possible values of the decision outcome itself, also known as the target variable. Therefore, the target variable is not included in the decision space, and consequently,

the sample space is not a subset of the decision space. There are three types of decision spaces relevant to the evaluation of the completeness of rule-based models, as follows.

An *attribute space* is a decision space that encompasses all potential combinations of values for a given set of attributes $(x_1, \dots, x_d)$ (Han et al., 2012). Therefore, the definition of an attribute space relies on the domain (Section 2.1.2.2) of the set of attributes. The domain ($\text{dom } x$) of an attribute represents the set of all its possible values (Fürnkranz et al., 2012). These values can be expressed as either a set, in the case of categorical variables (Equation 2.2) or an interval for numerical continuous variables (Equation 2.3). Formally, the attribute space of a set of attributes, denoted by $\mathcal{X}_A$, is defined as the Cartesian product of all individual attribute domains (Equation B.2).

$$\mathcal{X}_A = \text{dom } x_1 \times \dots \times \text{dom } x_d \quad (\text{B.2})$$

An *instance space* is a decision space derived from a dataset. While an attribute space is based on the domains of variables, an instance space is based on the empirical domains of variables, the set of observed values within the dataset (Section A.2.1). Similarly, an empirical domain ($\hat{\text{dom}} x$) of a variable can be expressed as either a set, in the case of categorical variables (Equation A.12) or an interval for numerical continuous variables (Equation A.13). Formally, the instance space of a dataset D , denoted by $\mathcal{X}_D$, is defined as the Cartesian product of the distinct observed values for all its variables (Equation B.3).

$$\mathcal{X}_D = \hat{\text{dom}} x_1 \times \dots \times \hat{\text{dom}} x_d \quad (\text{B.3})$$

A *learned space* is a decision space defined by the regions covered by a rule-based model (Sohns et al., 2022). A rule-based model encompasses R distinct rules, also referred to as a *rule-set*, denoted as $\mathcal{R}$ (Equation 2.33). Each rule can be expressed as an antecedent and consequent. The antecedent is a conjunction of d propositions applied to independent variables. The consequent represents the predicted class or value of the target variable (Weiss and Indurkha, 1995) (Equation 2.24). Each proposition within a rule defines a specific subset of possible values for its corresponding attribute (denoted as t_k). Collectively, these subsets form the edges of the rule's hyperrectangle, which represents the region of the decision space that the rule covers.

The hyperrectangle of a rule in the decision space, denoted as h_r (Equation 5.6), is defined by edges that encompass all condition subsets (t_k) associated with the variables tested in the rule (Equation B.4). The space enclosed by this hyperrectangle, denoted as $\mathcal{X}_{h_r}$, is defined by the Cartesian product of its condition subsets. Formally, the learned space of a model M , denoted by $\mathcal{X}_M$, is defined as the union of all hyperrectangles generated by the R rules within the model's rule set (Equation B.5).

$$\mathcal{X}_{h_r} = t_1 \times \dots \times t_d \quad (\text{B.4}) \quad \mathcal{X}_M = \mathcal{X}_{h_1} \cup \dots \cup \mathcal{X}_{h_R} \quad (\text{B.5})$$

Figure B.1 represents the scope of each decision space. In terms of cardinality, there is only one attribute space, as it is defined by the range of variables as concepts, being independent of datasets. However, there are as many instance spaces as there are datasets and as many learned

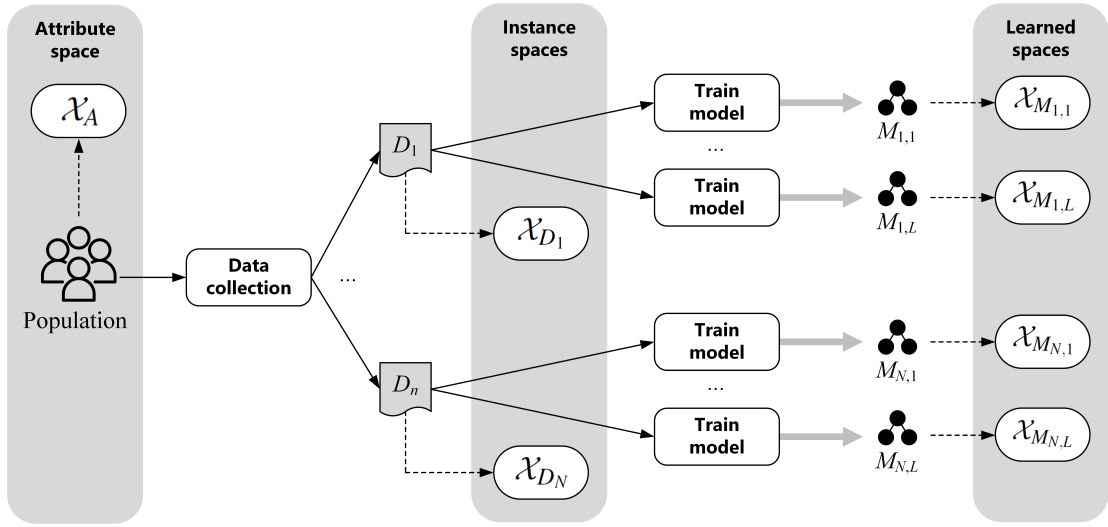


Figure B.1: A taxonomy of decision spaces

spaces as there are models. Consider a population characterized by attributes based on which decisions are made. The attribute space ($\mathcal{X}_A$) is defined by the combination of all values of these attributes. Within specific contexts, data is extracted into different datasets (D_1 to D_N). Each dataset captures specific observations of attribute values, and an instance space is defined for each dataset based on the values observed in that dataset ($\mathcal{X}_{D_1}$ to $\mathcal{X}_{D_N}$). Using the data from each dataset, different models ($M_{i,1}$ to $M_{i,L}, i \in \{1, \dots, N\}$) can be trained, and each model defines its own learned space ($\mathcal{X}_{M_{i,1}}$ to $\mathcal{X}_{M_{i,L}}, i \in \{1, \dots, N\}$).

To illustrate, consider a population characterized by the attributes of age (conceptually ranging from 0 to 120), gender (either ‘male’ or ‘female’), marital status (either ‘married’ or ‘not married’), and an outcome variable of income categorized into three levels (‘low’, ‘medium’, ‘high’). The attribute space would include all possible combinations of these attribute values, such as (20, ‘male’, ‘not married’), (35, ‘female’, ‘married’), and (65, ‘male’, ‘not married’). It is important to note that this space encompasses all combinations of attribute values, whether they make contextual sense or not, such as (3, ‘male’, ‘married’). If a dataset contains information only about individuals aged 20 to 30, the instance space will cover all possible combinations of age, gender, and marital status within this age range. When a decision tree model is trained on this dataset, the learned space represents the combinations of age, gender, and marital status that the model can classify into one of the income categories.

B.3 Completeness of rule-based models

Completeness is not a binary concept; instead, it can be regarded within a range encompassing:

- **Maximum completeness** – The model can predict the class for any incoming example within its instance space. This means it can handle any possible combination of values for the variables it considers.

- Partial completeness – The model can only predict the class for a subset of examples within the instance space. There might be some combinations of variable values for which it cannot make a prediction.
- Zero completeness – A model with zero completeness is a theoretical construct and not encountered in practice. Such a model would be devoid of any rules or decision-making logic.

To provide an overview of model completeness before resorting to formal measures, the following sections present two illustrative examples. The first example demonstrates how merging models can lead to an incomplete model. The second example focuses on the depictions of the different types of decision spaces, laying the groundwork for introducing the completeness metrics detailed in Section B.4.

B.3.1 Example: Incompleteness as a result of model merging

Figure B.2a depicts a dataset containing two numerical variables: x_1 and x_2 . Each data point represents an individual example within the dataset. These examples are categorized into two classes, c_1 and c_2 , illustrated by filled and empty circles, respectively. The observed values in all data points for x_1 and x_2 define their respective empirical domains, which define, in turn, the instance space ($\mathcal{X}_D$) as follows:

$$\begin{aligned}\hat{\text{dom}} x_1 &= [0; a_2] \\ \hat{\text{dom}} x_2 &= [0; b_4] \\ \mathcal{X}_D &= [0; a_2] \times [0; b_4]\end{aligned}$$

Subsequently, two rule-based models, M_A and M_B , were trained from this dataset (Figures B.2b and B.2c). The different shadings in each region represent the predicted classes by each rule, with rule sets formally specified as follows:

$$\begin{aligned}\mathcal{R}_A &= \begin{cases} r_1 : (x_1 \in [0; a_1]) \rightarrow c_1 \\ r_2 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [b_3; +\infty)) \rightarrow c_1 \\ r_3 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [0; b_3)) \rightarrow c_2 \end{cases} \\ \mathcal{R}_B &= \begin{cases} r_1 : (x_1 \in [0; a_1] \wedge x_2 \in [b_2; +\infty)) \rightarrow c_2 \\ r_2 : (x_1 \in [0; a_1] \wedge x_2 \in [0; b_2)) \rightarrow c_1 \\ r_3 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [b_1; +\infty)) \rightarrow c_1 \\ r_4 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [0; b_1)) \rightarrow c_2 \end{cases}\end{aligned}$$

Both models have maximum completeness as the underlying rules cover the entire instance space. Following the analysis of the individual models, a merged model, M_C , can be developed

through an intersection operation (Figure B.2d). This operation focuses exclusively on the regions where both models agree on the predicted class (i.e., without conflicts), with the rule-set as follows:

$$\mathcal{R}_C = \begin{cases} r_1 : (x_1 \in [0; a_1) \wedge x_2 \in [0; b_2)) \rightarrow c_1 \\ r_2 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [b_3; +\infty)) \rightarrow c_1 \\ r_3 : (x_1 \in [a_1; +\infty) \wedge x_2 \in [0; b_1)) \rightarrow c_2 \end{cases}$$

Model M_C is incomplete as there are regions within the instance space that are not covered by any of the model's rules. For example, a data point with coordinates $x_1 = a_1/2$ and $x_2 = b_3$ would not fall within any hyperrectangle, consequently not triggering any rules.

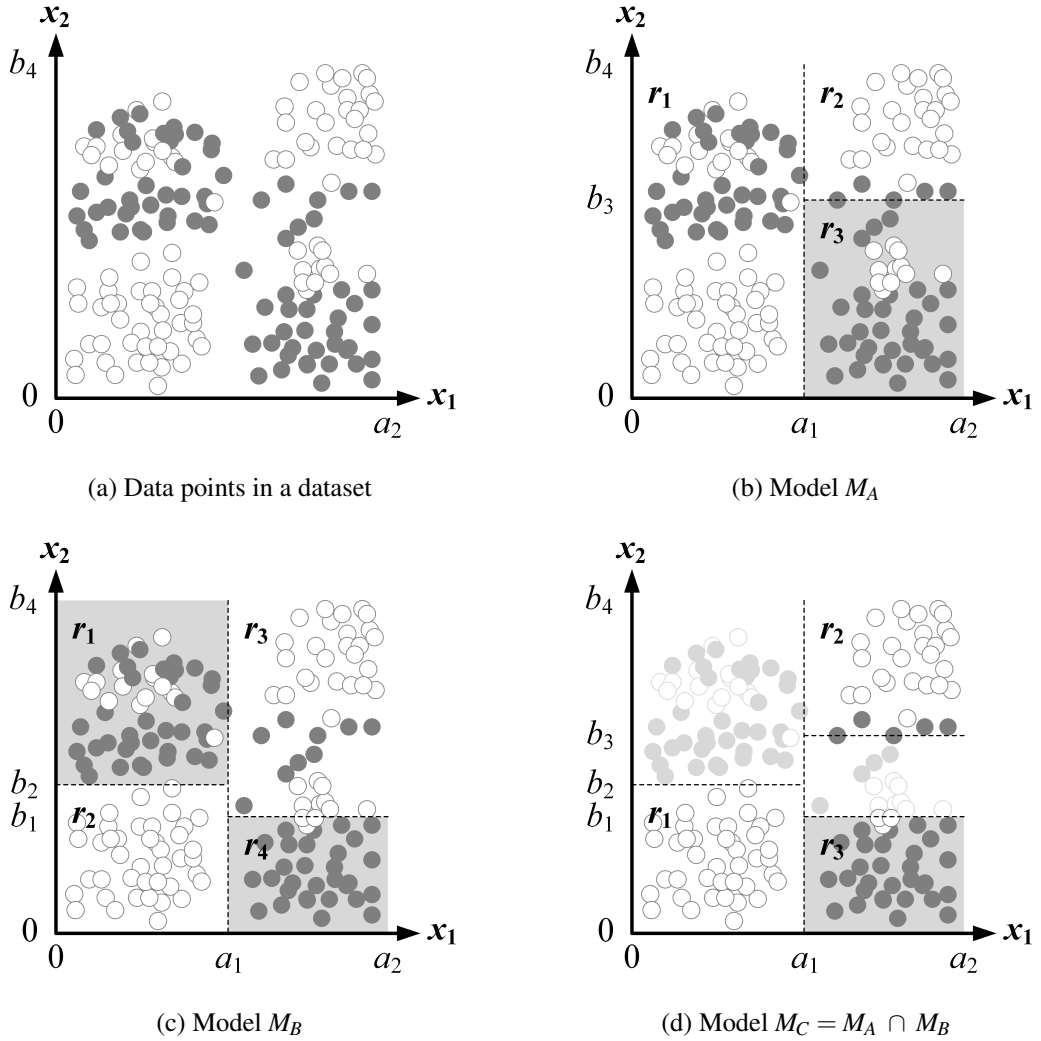


Figure B.2: Geometric depiction of model intersection coverage

B.3.2 Example: Completeness in the context of decision spaces

Figure B.3 provides a visual representation of the three types of decision spaces. The variables x_1 (numerical), x_2 (categorical), and x_3 (numerical) define an attribute space ($\mathcal{X}_A$) from a domain-knowledge perspective. In other words, this definition is based on business concepts and does not rely on a specific dataset. The domains and attributes space are defined as follows:

$$\begin{aligned}\text{dom } x_1 &= [a_1; a_6] \\ \text{dom } x_2 &= \{b_1, b_2, b_3, b_4\} \\ \text{dom } x_3 &= [d_1; d_3] \\ \mathcal{X}_A &= [a_1; a_6] \times \{b_1, b_2, b_3, b_4\} \times [d_1; d_3]\end{aligned}$$

Once data collection for a dataset D is complete, the observed values of the variables define the empirical domains. These empirical domains, in turn, define an instance space ($\mathcal{X}_D$), as follows:

$$\begin{aligned}\hat{\text{dom}} x_1 &= [a_1; a_5] \\ \hat{\text{dom}} x_2 &= \{b_1, b_2, b_3\} \\ \hat{\text{dom}} x_3 &= [d_1; d_3] \\ \mathcal{X}_D &= [a_1; a_5] \times \{b_1, b_2, b_3\} \times [d_1; d_3]\end{aligned}$$

A rule-based model M is then trained from the dataset, comprising three rules (r_1 , r_2 , and r_3), with the rule-set formally specified as follows:

$$\mathcal{R} = \begin{cases} r_1 : (x_1 \in [a_1; a_2] \wedge x_2 \in \{b_1, b_2\} \wedge x_3 \in [d_1; d_3]) \rightarrow c_1 \\ r_2 : (x_1 \in [a_3; a_4] \wedge x_2 \in \{b_1, b_2\} \wedge x_3 \in [d_1; d_3]) \rightarrow c_2 \\ r_3 : (x_1 \in [a_4; a_5] \wedge x_3 \in [d_2; d_3]) \rightarrow c_2 \end{cases}$$

Each rule (r_i) specifies a hyperrectangle (h_i) within the instance space using subsets (t) that define the test conditions for each variable. Each hyperrectangle, in turn, encloses a space ($\mathcal{X}_{h_i}$), defined by the cross product of the subsets.

$$\begin{aligned}r_i &: (x_1 \in t_{i,1} \wedge x_2 \in t_{i,2} \wedge x_3 \in t_{i,3}) \rightarrow c_i \\ h_i &= (t_{i,1}, t_{i,2}, t_{i,3}) \\ \mathcal{X}_{h_i} &= t_{i,1} \times t_{i,2} \times t_{i,3}\end{aligned}$$

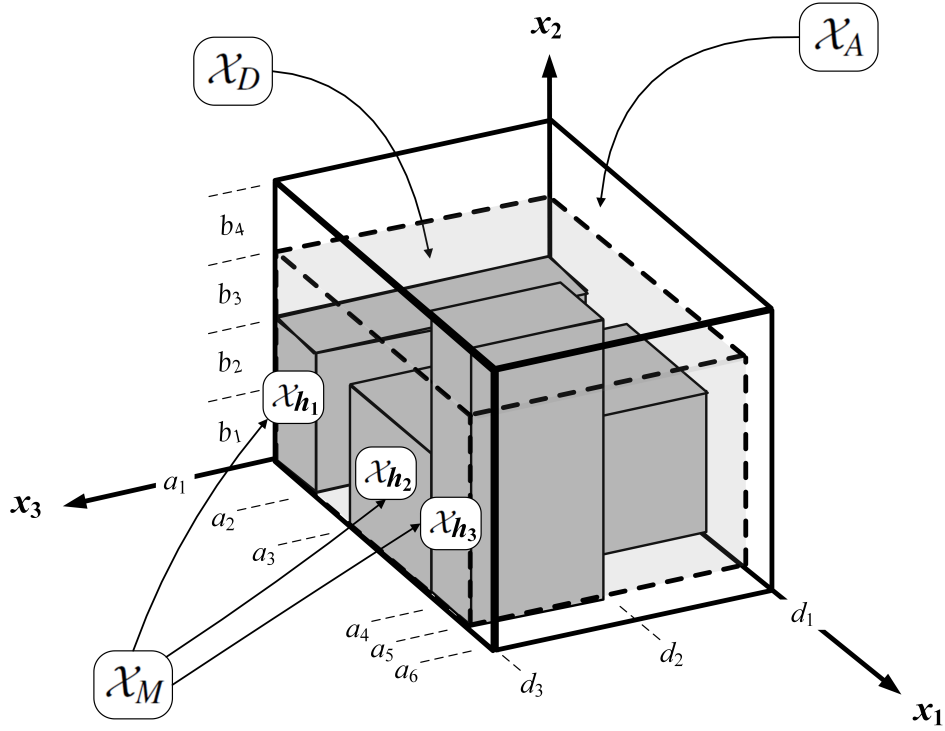


Figure B.3: An attribute space, an instance space, and a learned space

The learned space ($\mathcal{X}_M$) is defined from the union of the spaces associated with three hyperrectangles, as follows:

$$\mathcal{X}_{h_1} = [a_1; a_2] \times \{b_1, b_2\} \times [d_1; d_3]$$

$$\mathcal{X}_{h_2} = [a_3; a_4] \times \{b_1, b_2\} \times [d_1; d_3]$$

$$\mathcal{X}_{h_3} = [a_4; a_5] \times [d_2; d_3]$$

$$\mathcal{X}_M = \mathcal{X}_{h_1} \cup \mathcal{X}_{h_2} \cup \mathcal{X}_{h_3}$$

It is noteworthy that rule r_3 lacks a test condition for variable x_2 . Consequently, its test value for x_2 is assumed to be the entire domain ($t_{3,2} = \text{dom } x_2$). Due to this missing test condition, $\mathcal{X}_{h_3}$ extends beyond the instance space. This extension results from the model's generalization over variable x_2 . However, the model remains incomplete relative to both the attribute and instance spaces. This incompleteness arises because the model cannot make predictions for examples located between the space of hyperrectangles h_1 and h_2 .

B.4 Metrics to measure completeness of rule-based models

There are two potential reference spaces to evaluate the completeness of a model: the attribute space and the instance space. While the attribute space encompasses all attribute-value possibilities, it demands a comprehensive understanding of the underlying business rules. As acquiring

such domain knowledge may not always be feasible, completeness is typically measured with reference to the instance space, which is automatically derived from data.

Fraction Covered (Section 3.1.4.2), though seemingly relevant for evaluating the completeness of rule-based models, relies on the availability of a test set. While it can be used to evaluate the completeness relative of a consensus model, it is important to consider metrics that are not dependent on specific test sets. To address this limitation, two novel approaches are introduced: geometrical and distributional methods, both designed to evaluate the completeness of rule-based models, detailed in the next sections.

B.4.1 Geometrical completeness

The *geometrical completeness* measures the learned space in relation to the instance space. The instance space is represented as a single hyperrectangle, allowing for the calculation of its hypervolume. In contrast, the hypervolume of the learned space is determined by adding the hypervolumes of the hyperrectangles associated with each rule in the model (Equation B.6).

Formally, the geometrical completeness of a model, denoted by $GComp$, is defined as the ratio between the hypervolume of the learned space ($V_{\mathcal{X}_M}$) and the hypervolume of the instance space ($V_{\mathcal{X}_D}$) (Equation B.7). This calculation assumes that the instance space completely encompasses the learned space. Therefore, maximum completeness is achieved when the learned space is equal to the instance space.

$$V_{\mathcal{X}_M} = \sum_{i=1}^R V_{h_i} \quad (\text{B.6})$$

$$GComp = \frac{V_{\mathcal{X}_M}}{V_{\mathcal{X}_D}} \quad (\text{B.7})$$

Figure B.4 builds upon Figure B.3 by depicting model M within the context of the instance space. This figure focuses on the empirical domains observed for the variables within a specific dataset used to train the model, as follows:

$$\begin{aligned} \hat{\text{dom}}_{x_1} &= [0; 9] \\ \hat{\text{dom}}_{x_2} &= \{e, g, p\} \\ \hat{\text{dom}}_{x_3} &= [0; 20] \end{aligned}$$

Calculating the hypervolume of each rule's hyperrectangle requires considering the width of each edge (Equation A.22). For numerical variables (discrete or continuous), the width is the difference between the upper and lower bounds of the edge (Equation A.20). For categorical variables, the width is the number of distinct values within the edge (Equation A.21). In this context, an edge aligns with the formal definition of the variable's subset within the hyperrectangle. To measure the hypervolume of the instance space, it is assumed that each edge corresponds to

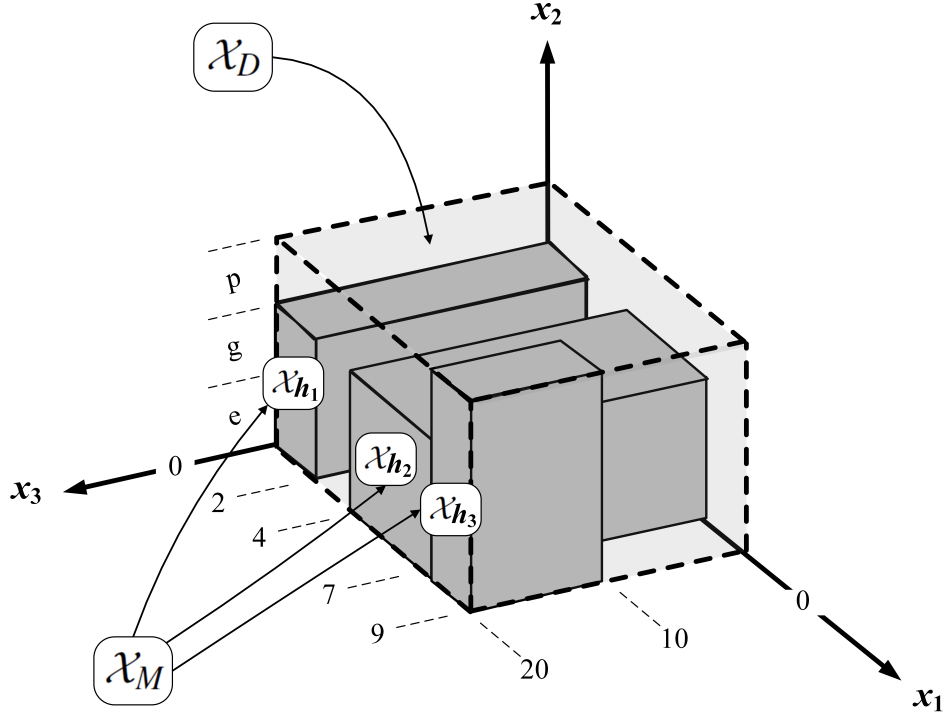


Figure B.4: A learned space within an instance space

the empirical domain of its respective variable. The width of each edge in the instance space is determined similarly to the rule hyperrectangles. Based on this analysis, the model's geometrical completeness is calculated to be 0.55, indicating that it covers approximately half of the instance space.

$$\begin{aligned}
 \mathcal{X}_{h_1} &= [0;2) \times \{e, g\} \times [0;20) \\
 \mathcal{X}_{h_2} &= [3;7) \times \{e, g\} \times [0;20) \\
 \mathcal{X}_{h_3} &= [7;9) \times \{e, g, p\} \times [10;20) \\
 \mathcal{X}_D &= [0;9) \times \{e, g, p\} \times [0;20) \\
 V_{h_1} &= w_{1,1} \times w_{2,1} \times w_{3,1} = 2 \times 2 \times 20 = 80 \\
 V_{h_2} &= w_{1,2} \times w_{2,2} \times w_{3,2} = 4 \times 2 \times 20 = 160 \\
 V_{h_3} &= w_{1,3} \times w_{2,3} \times w_{3,3} = 2 \times 3 \times 10 = 60 \\
 V_{\mathcal{X}_D} &= |\hat{\text{dom}} x_1| \times |\hat{\text{dom}} x_2| \times |\hat{\text{dom}} x_3| = 9 \times 3 \times 20 = 540
 \end{aligned}$$

$$GComp = \frac{300}{540} = 0.55$$

B.4.2 Distributional completeness

The *distributional completeness* measures the proportion of data covered by the learned space relative to the underlying data distribution within the instance space. This approach first defines a hypergrid encompassing the entire instance space. Each hyperrectangle of the model’s rules can then fully enclose one or more hyperrectangles of the hypergrid without any partial overlap.

Next, a multivariate density estimator, denoted by $\Gamma_{\hat{p}}$, is created based on the same dataset used to define the instance space. This estimator captures the density distribution across the instance space in K hyperrectangles. A subset of $\Gamma_{\hat{p}}$, denoted by $\Gamma_{\hat{p}_M}$, is then assembled by including only the grid’s hyperrectangles that are entirely contained within the hyperrectangles of the model’s rules (Equation B.8).

$$\Gamma_{\hat{p}_M} = \{h_k : h_k \subseteq \mathcal{X}_M \wedge k \in [1, K]\} \quad (\text{B.8})$$

Formally, the distributional completeness of a model, denoted by $DComp$, is defined as the weighted sum of the density estimates for the hyperrectangles enclosed by the model’s rules. The weights correspond to the hypervolumes of these respective hyperrectangles (Equation B.9). Maximum completeness is achieved when the learned space fully encloses all hyperrectangles with a density greater than zero.

$$DComp = \sum_{k=1}^{|\Gamma_{\hat{p}_M}|} V_k \hat{p}_k \quad (\text{B.9})$$

To illustrate, Figure B.5 depicts a hypergrid with granularity aligned to the edges of the hyperrectangles corresponding to the rules of model M (introduced in Figure B.3). The minimum granularity required for the hypergrid to measure the distributional completeness of M is restricted by the edges of its hyperrectangles. For this particular case, the hypergrid defines hyperrectangles for each integer value of variable x_1 , each category of variable x_2 , and every 10-unit interval of variable x_3 . This results in a total of 54 hyperrectangles within the hypergrid. Figure B.6 depicts a subset of the hypergrid consisting exclusively of hyperrectangles from the hypergrid that are entirely enclosed within the learned space of model M .

Table B.1 exemplifies a multivariate density estimator ($\Gamma_{\hat{p}}$) from a specific dataset, with each row corresponding to a hyperrectangle of the grid. The bin information (b) refers to the specific range for each variable within the hyperrectangle. The example count (N) represents the number of examples that fall within that specific hyperrectangle. Density estimation ($\hat{p}$) denotes the data density within that particular hyperrectangle; to obtain the probability, the density must be multiplied by the hypervolume of the hyperrectangle. In this example, both the x_1 and the x_2 have a width of 1 (unitary edges). Therefore, for all hyperrectangles, their hypervolume is equivalent to the width of x_3 ($w_3 = 10$, in all hyperrectangles).

Table B.2 presents a subset of hyperrectangles from the density estimator ($\Gamma_{\hat{p}_M}$), which are all entirely enclosed within the model’s learned space, in a total of 30 hyperrectangles. The table specifies the hyperrectangle in the learned space for index j , along with its corresponding entry in

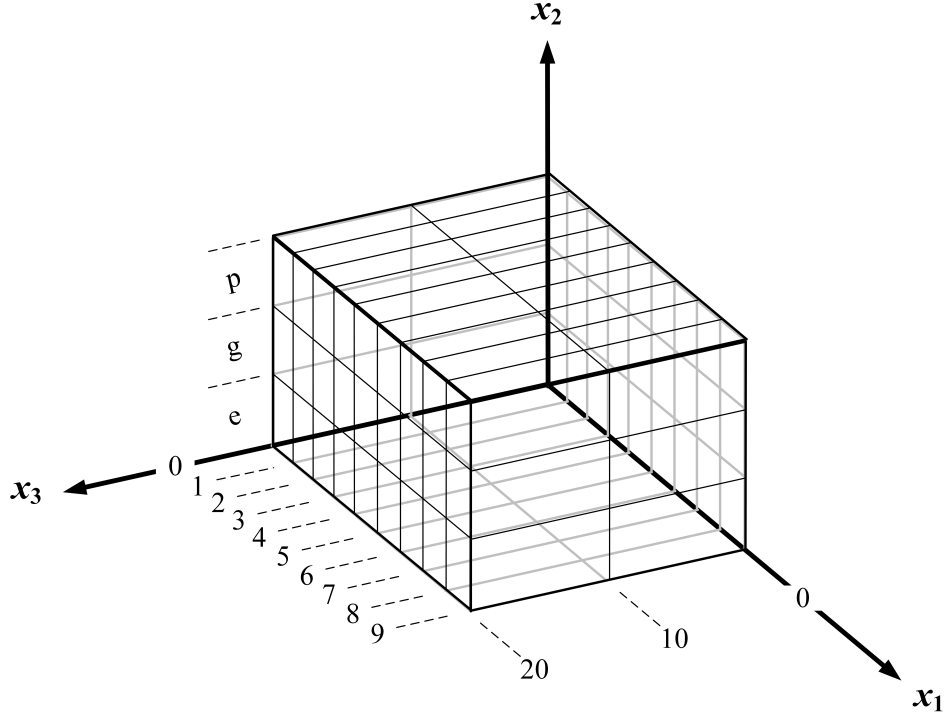


Figure B.5: A hypergrid as a basis for a density estimator

Table B.1 in index k . It also indicates the product of the hypervolume and the density estimation. The sum of this last column reveals that the model's distributional completeness is 0.87. This value conveys that, based on the underlying data distribution, the model can make predictions for nearly 90% of the data.

B.5 Evaluation on selected datasets

B.5.1 Experimental setup

The *Inmplode* framework incorporates completeness as a evaluation criteria alongside others, such as predictive performance, interpretability, and fidelity. To compare the results of the two metrics of completeness, the merging process was applied to five datasets described in Table 7.1 with details presented in Appendix C.

Both geometrical and distributional completeness require an estimate of data density, represented by a multivariate density estimator, presented in Appendix A, denoted by $\Gamma_{\hat{p}}$, derived from each dataset. Following the generic workflow of the *Inmplode* framework, the data from N business units results in datasets $D_1, \dots, D_N$. A decision tree model $(M_1, \dots, M_N)$ is trained on each of these unit-specific datasets. After training, the individual models are then grouped into strata $(S_1, \dots, S_J)$ based on specific criteria. Within each stratum, the models are sequentially converted into decision tables and merged to yield a final decision table. Therefore, for each stratum, there is a corresponding table resulting from the merging of all models within that stratum.

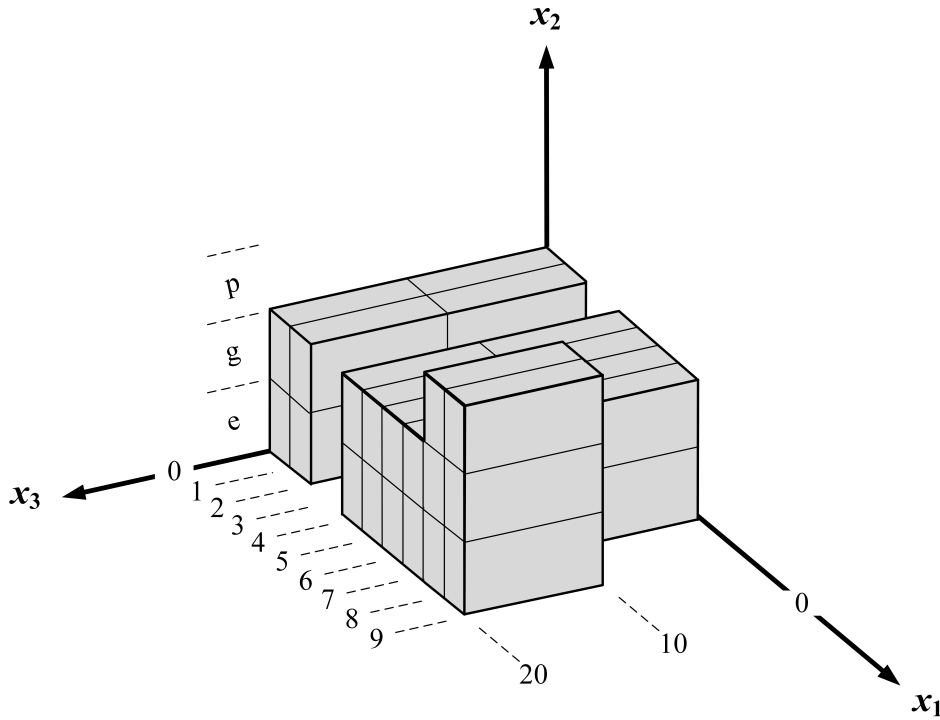


Figure B.6: A learned space within a hypergrid

The process of building a consensus model was performed by merging models by intersection. The evaluation of geometrical and distributional completeness was carried out while merging each table using the previously derived multivariate histogram (Γ) from the dataset (Figure B.7). The instance space required for geometrical completeness was obtained from the multivariate density estimator. The completeness evaluation process generates a tuple, denoted as Z_i , which holds the completeness results associated with the merge attempt $\bar{T}_i$.

B.5.2 Results and discussion

B.5.2.1 Completeness results within strata

The consensus model-building process involves sequentially merging operational models. This allows for monitoring the completeness progression (both geometrical and distributional) across different strata within select datasets listed in Table 7.1. Figures B.8 to B.12 depict the results for completeness as more decision tables from each stratum are merged within setup #1.

In the case of the University of Porto, Flight Delays, and Adult Census datasets, the analysis focuses on the initial 11 operational models, resulting in a total of 10 attempts to merge models. In contrast, for the Bank Marketing and Hotel Reservations datasets, each consensus model corresponds to a semester, therefore representing six operational models. The attempt model combines data from the first and second months, while the fifth attempt integrates data from the fourth merged model and the model of the sixth month.

Table B.1: Multivariate density estimator ($\Gamma_{\hat{p}}$)

k	b_1	b_2	b_3	N	$\hat{p}$
1	(0; 1]	{e}	(0; 10]	5	0.0005
2	(0; 1]	{e}	(10; 20]	19	0.0019
3	(0; 1]	{g}	(0; 10]	13	0.0013
4	(0; 1]	{g}	(10; 20]	45	0.0045
5	(0; 1]	{p}	(0; 10]	5	0.0047
6	(0; 1]	{p}	(10; 20]	3	0.0022
7	(1; 2]	{e}	(10; 20]	47	0.0047
8	(1; 2]	{e}	(10; 20]	22	0.0022
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
53	(8; 9]	{p}	(0; 10]	2	0.0002
54	(8; 9]	{p}	(10; 20]	38	0.0038

Analysis across various datasets reveals a consistent pattern: both geometrical and distributional completeness metrics tend to decrease as the number of operational models integrated into a consensus model increases. This trend is intuitive. Initially, when merging similar models, the completeness metrics start near or at one, signifying a high degree of completeness since the models capture similar aspects of the instance space. However, as increasingly dissimilar models are incorporated into the consensus, a decline in completeness becomes clear. These dissimilar models may introduce conflicting information or cover different regions of the instance space, leading to a less comprehensive representation of the overall data distribution.

Geometrical completeness consistently falls below distributional completeness in all datasets. This suggests the models generally cover a smaller portion of the instance space compared to the overall data distribution. Analyzing individual datasets reveals distinct trends. In the University of Porto and Bank Marketing datasets, geometrical completeness exhibits a steeper decline, deviating significantly from distributional completeness. This rapid decrease implies that incorporating diverse models quickly reduces the models' coverage of the instance space. Both metrics eventually converge towards zero as the final model merging is reached. In contrast, the Flight Delays, Adult Census, and Hotel Reservations datasets showcase a correlation between the two metrics, where geometrical completeness remains slightly below distributional completeness throughout the process. This indicates that incorporating diverse models in these cases leads to a more gradual decrease in both measures, suggesting a smoother decline in coverage of both the instance space and the data distribution.

The analysis reveals a pattern regarding model similarity and decision space coverage. When models share a high degree of similarity, their intersection (the merged coverage area) effectively maintains the overall coverage of the instance space. In contrast, the intersection of highly dissimilar models results in a permanent reduction in coverage. This is because the overlapping area

Table B.2: Multivariate density estimator restricted to the learned space ($\Gamma_{\hat{\rho}_M}$)

j	k	b_1	b_2	b_3	$V \times \hat{p}$
1	1	(0;1]	{e}	(0;10]	0.005
2	2	(0;1]	{e}	(10;20]	0.019
3	3	(0;1]	{g}	(0;10]	0.013
4	4	(0;1]	{g}	(10;20]	0.045
5	7	(1;2]	{e}	(10;20]	0.047
6	8	(1;2]	{e}	(10;20]	0.022
7	9	(1;2]	{g}	(10;20]	0.002
8	10	(1;2]	{g}	(10;20]	0.037
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
29	52	(8;9]	{g}	(10;20]	0.046
30	54	(8;9]	{p}	(10;20]	0.038
$DComp =$					0.870

between them represents the data points they can both agree on, which is likely a smaller portion of the total space.

As more models are integrated into the consensus model, geometrical completeness tends to decrease gradually. This is likely due to incorporating diverse models that can introduce conflicting information or cover different regions of the instance space, leading to less comprehensive overall coverage. In extreme cases where highly dissimilar models are included, geometrical completeness can rapidly approach zero, indicating minimal overlap in coverage between the merged models and the entire instance space.

Distributional completeness, which considers data concentration, remains unaffected by decision space coverage. Decision trees tend to generate rules based on data-dense regions and subsequently extend these rules to sparser areas. As a consequence, distributional completeness often exhibits higher values when compared to geometrical completeness, as the latter measures all regions of the decision space uniformly. This fundamental distinction plays a critical role in accounting for the observed disparities between the two completeness metrics.

B.5.2.2 Completeness results across setups

Figures B.13 to B.17 depict the distribution of geometrical and distributional completeness scores for all strata within each dataset across setups #1, #2, and #3 (out of a total of 10 setups). Appendix C includes the results across all setups.

Both completeness metrics exhibit a rapid decline toward zero in the University of Porto and Flight Delays datasets. This suggests a significant difference between the models within each stratum, leading to a limited overall coverage of the instance space (geometrical completeness) and a mismatch with the data distribution (distributional completeness). In contrast, the Adult

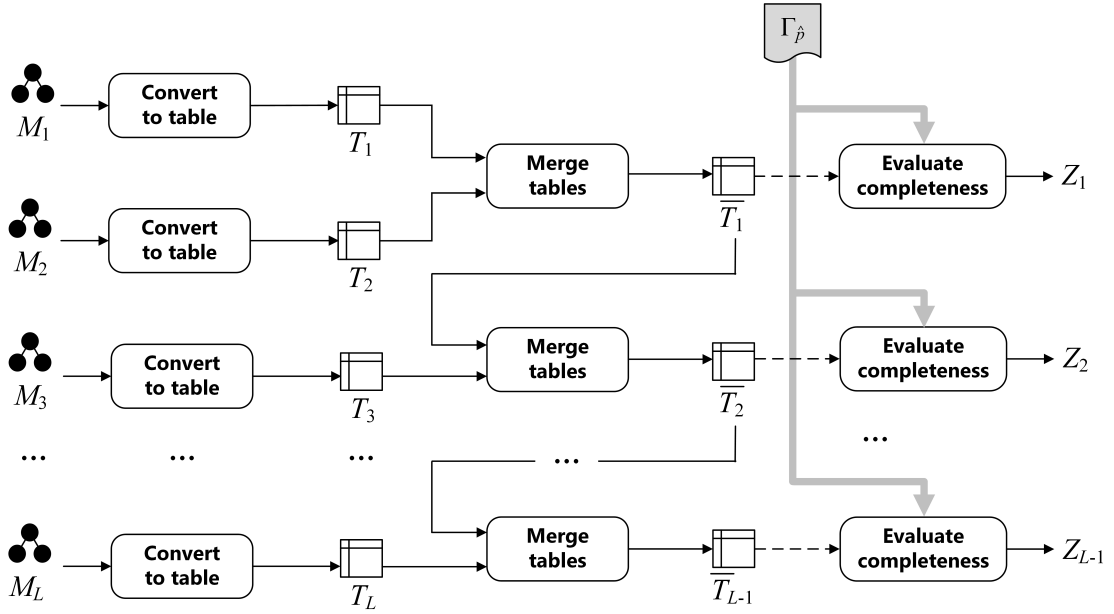


Figure B.7: Evaluating completeness of merged decision tables

Census dataset shows consistently high values for both metrics. This indicates a high degree of similarity between the models within each stratum, resulting in a good coverage of the instance space and a strong alignment with the data distribution.

The Bank Marketing and Hotel Reservations datasets showcase a distinct pattern. Initially, both metrics exhibit high values, suggesting substantial similarity among the first models incorporated into the consensus. However, geometrical completeness shows a steeper decline in later strata, indicating that subsequently added models become increasingly dissimilar, leading to a decrease in overall instance space coverage. Distributional completeness, on the other hand, exhibits a more gradual decrease, suggesting some alignment with the data distribution remains even with these dissimilar models.

These observations suggest that beyond the specific datasets analyzed, there is a relationship between:

- Geometrical completeness and decision space coverage – The rapid decline in geometrical completeness across most datasets implies that incorporating increasingly dissimilar models leads to a significant decrease in overall decision space coverage.
- Distributional completeness and data distribution coverage – In contrast, the persistence of non-zero values for distributional completeness, even when geometrical completeness drops, indicates an alignment with the data distribution.

This distinction between geometrical and distributional completeness highlights the importance of considering both metrics when evaluating the completeness of a consensus model. In practical applications, the relative importance of each metric may depend on the specific task. For example, if accurate classification of all data points is critical, a high geometrical completeness

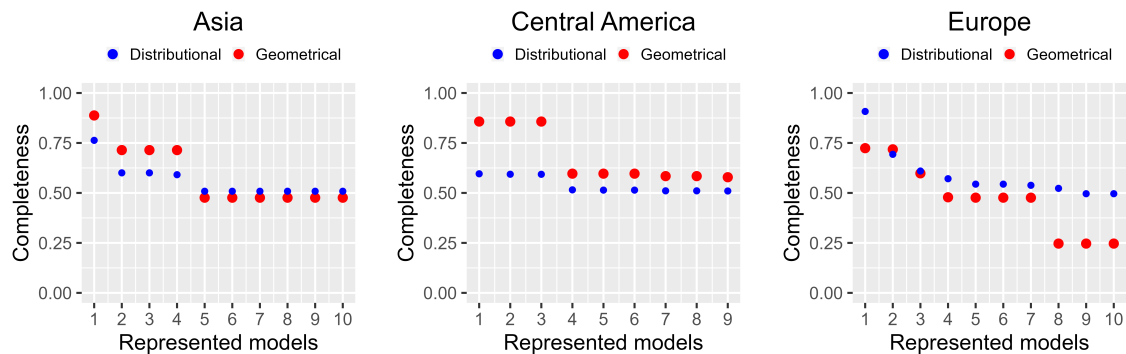


Figure B.8: Adult Census dataset (setup #1): Completeness in merged decision tables

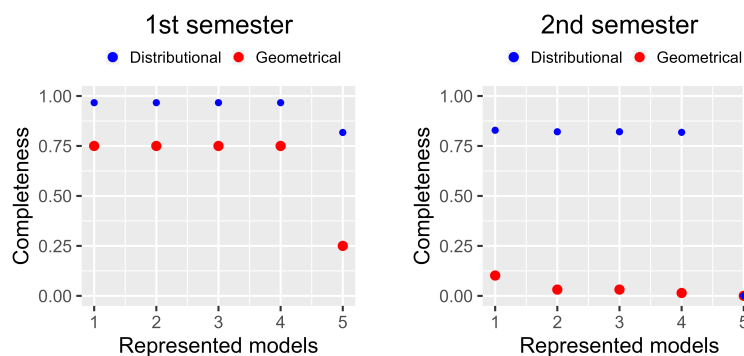


Figure B.9: Bank Marketing dataset (setup #1): Completeness in merged decision tables

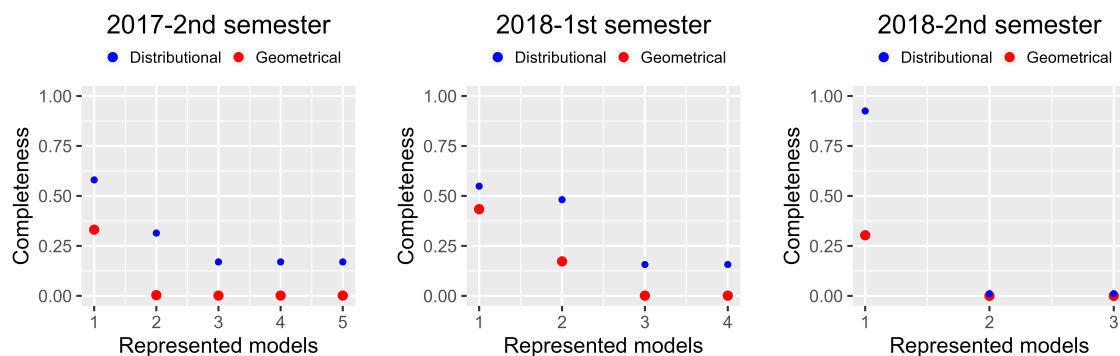


Figure B.10: Hotel Reservations dataset (setup #1): Completeness in merged decision table

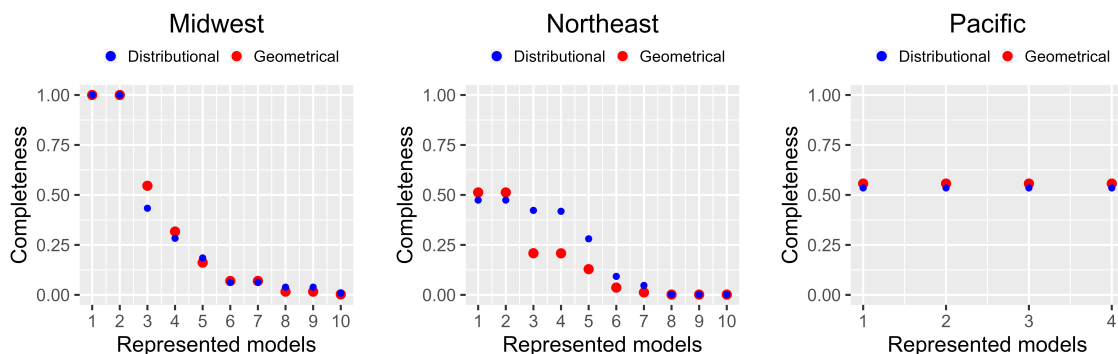


Figure B.11: Flight Delays dataset (setup #1): Completeness in merged decision tables

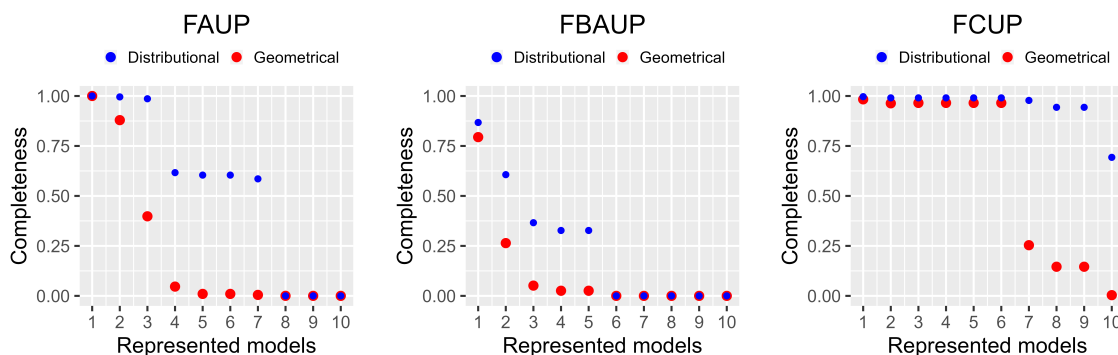


Figure B.12: University of Porto dataset (setup #1): Completeness in merged decision tables

may be essential. However, if the goal is to capture the overall trends and patterns within the data, a moderate level of geometrical completeness might be acceptable, especially if distributional completeness remains high.

B.6 Summary

Rule-based models are generally considered complete due to their inherent properties that ensure comprehensive coverage of decision scenarios. Decision trees, for instance, exhibit exhaustiveness, encompassing the entire decision space within their rules. Similarly, classification rules typically incorporate a default rule for instances not matched by other rules. However, the issue of measuring completeness becomes pertinent in scenarios where rule-based models are derived from merging, such as in the creation of consensus models. This process can lead to diminished coverage of the decision space when rules are derived from the intersection of multiple models.

While individual rule-based models may exhibit completeness, the merging process can introduce geometrical incompleteness in consensus models. Empirical observations indicate that geometrical completeness often approaches zero rapidly after combining just a few models. This implies that the merged model may not effectively encompass all regions of the decision space.

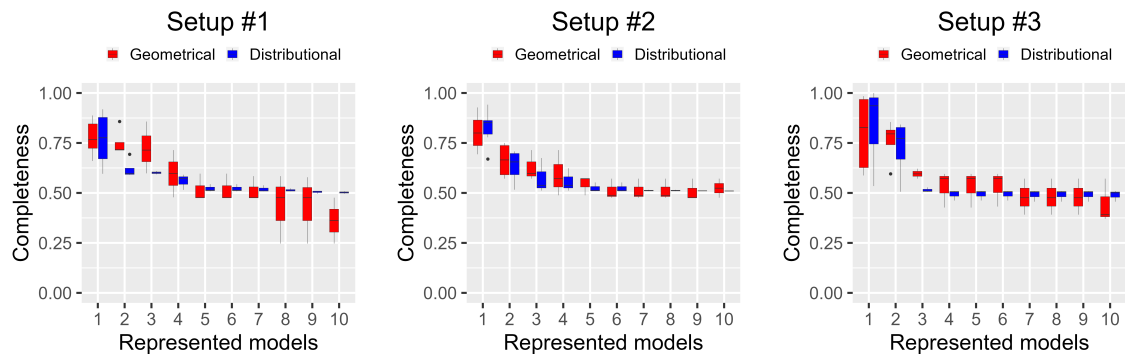


Figure B.13: Adult Census dataset: Completeness in merged decision tables in setups

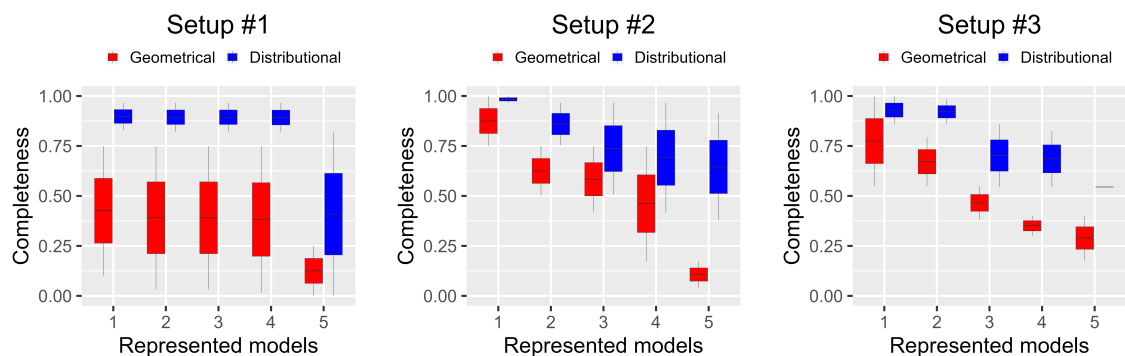


Figure B.14: Bank Marketing dataset: Completeness in merged decision tables in setups

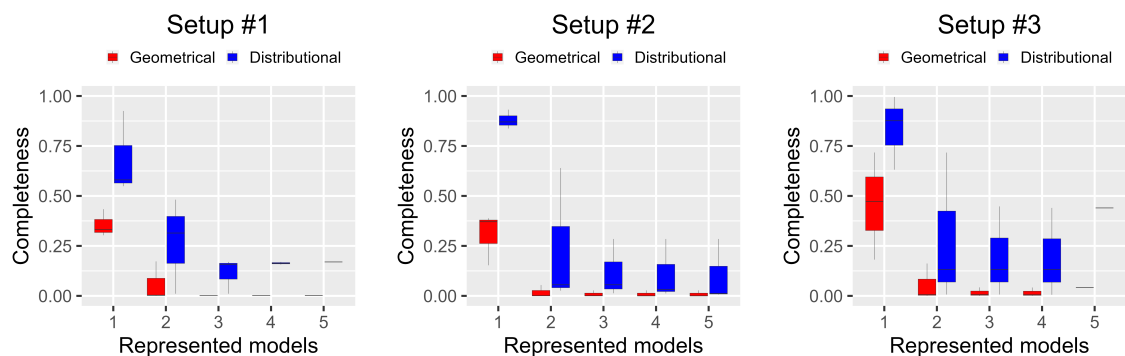


Figure B.15: Hotel Reservations dataset: Completeness in merged decision tables in setups

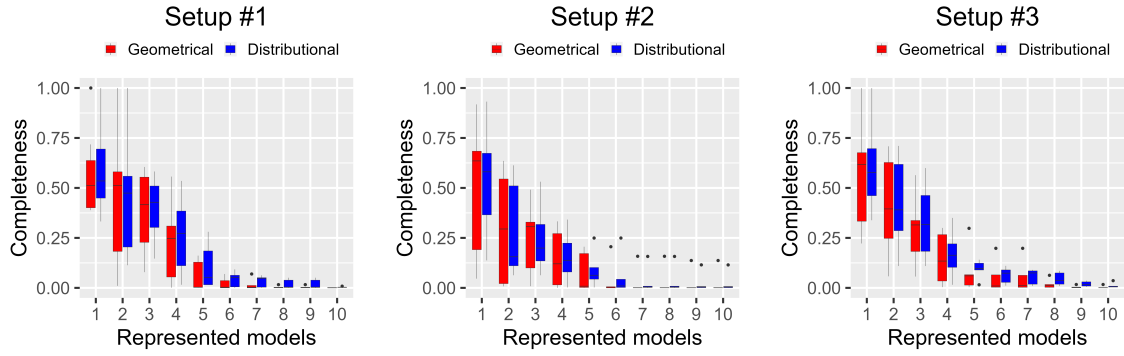


Figure B.16: Flight Delays dataset: Completeness in merged decision tables in setups

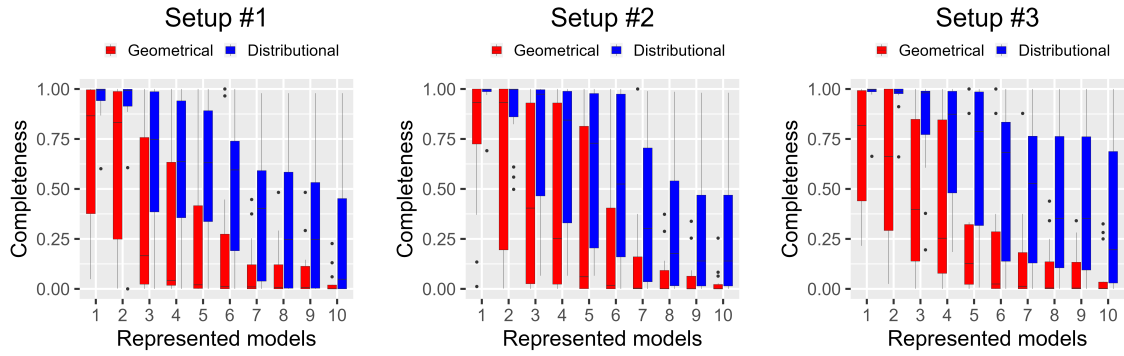


Figure B.17: University of Porto dataset: Completeness in merged decision tables in setups

Conversely, distributional completeness demonstrates a more gradual decline, suggesting that consensus models tend to retain alignment with the underlying data distribution, even as geometrical completeness diminishes. This alignment with data density renders consensus models suitable for practical applications.

These findings underscore both the limitations and strengths of the model merging approach. Consensus models can effectively capture regions of high data concentration, yet they may not provide comprehensive coverage of the entire decision space. This observation provides context for investigating completeness measures and developing strategies to mitigate completeness loss, thereby enhancing the applicability and robustness of consensus models in real-world scenarios.

Appendix C

Datasets, Models and Evaluation Results

This appendix provides detailed information regarding the datasets used in the experiments for creating operational models and subsequently developing consensus models through the application of the *Inmplode* framework. For each dataset, a description of the types of examples (instances) and variables (attributes) is presented. This is followed by a presentation of the business units that form the basis for the creation of operational models, along with examples from each dataset. Additionally, the stratification criteria and the developed consensus models are outlined. The results, presented in Chapter 7, are detailed in this appendix in all three representations of consensus models: merged tables, synthetic data trees, and distributional data trees. The source code for *Inmplode* is available at <https://github.com/pstrech/inmplode>.

C.1 Datasets description

Table C.1 provides a summary of the five datasets used in the experiments. These include four publicly available datasets from the Kaggle repository (Kaggle, 2024), namely, Adult Census, Bank Marketing, Flight Delays, Hotel Reservations. In addition, a dataset obtained from the University of Porto’s academic information system remains private due to confidentiality concerns. Further details on each dataset are provided in subsequent sections.

Table C.1: Datasets summary

Dataset	n	d	Examples	Prediction
Adult Census	49K	12	People	Income level
Bank Marketing	45K	15	Clients	Subscribed campaign
Hotel Reservations	36K	18	Reservations	Canceled reservation
Flight Delays	539K	8	Flights	Delayed flight
University of Porto	1300K	26	Enrollments	Failed course

The selected datasets provide a robust and diverse foundation for training classification models. Their varying sizes (n), number of independent variables (d), business areas, sources, and potential stratification criteria (e.g., time-based or geographical) enhance the generalizability of results across different real-world scenarios. The inclusion of publicly available datasets, widely recognized within the machine learning community, allows for direct comparison with existing research. Additionally, the University of Porto dataset, designed for decision support systems, introduces academic-based stratification options (faculties, courses, scientific areas), further expanding the potential applications of the developed framework. This combination of publicly available and domain-specific datasets ensures a comprehensive evaluation of the framework’s performance and applicability in diverse contexts.

In the following sections, the variable types for each dataset are described. Unique identifiers were removed prior to model training. The variable to be predicted is denoted as y , with all other variables considered independent. Some datasets were used in their original form, while others were modified (e.g., by removing columns) to better suit the specific analysis.

C.1.1 Adult Census dataset

The Adult Census dataset, extracted from the 1994 Census Bureau database (Kohavi, 1996), is used for predicting whether an individual’s annual income exceeds 50000 dollars based on demographic and employment data. The dataset comprises 48842 examples (individuals) described by the variables in Table C.2. Missing values in categorical variables were replaced with the label ‘unknown’. Additionally, the ‘fnlwgt’ variable (final weight) and the ‘education-num’ variable (a numerical representation of education level) were excluded from the analysis due to irrelevance to the prediction task and redundancy, respectively.

Table C.2: Description of variables of the Adult Census dataset

Variable	Description	Type
x_1 = Age	Age of the individual	Numerical discrete
x_2 = Workclass	Type of employment	Categorical
x_3 = Education	Level of education completed	Categorical
x_4 = Marital status	Marital status of the individual	Categorical
x_5 = Occupation	Type of occupation	Categorical
x_6 = Relationship	Relationship status within the household	Categorical
x_7 = Race	Race of the individual	Categorical
x_8 = Sex	Gender of the individual	Categorical
x_9 = Capital gain	Income from investment sources	Numerical discrete
x_{10} = Capital loss	Losses from investment sources	Numerical discrete
x_{11} = Hours per week	Number of hours worked per week	Numerical discrete
x_{12} = Native country	Country of origin	Categorical
y = Income	Income category ($<50K$ or $\geq 50K$)	Categorical

C.1.2 Bank Marketing dataset

The Bank Marketing dataset offers a comprehensive record of a Portuguese bank’s direct marketing campaigns (phone calls), including customer demographics, interaction details, and campaign outcomes. This data, detailed in Table C.3, provides a resource for understanding customer behavior and optimizing future marketing strategies. By analyzing patterns within the 45211 client records (which contain no missing values), the aim is to identify factors influencing customer responses to term deposit subscription offers. This analysis informs actionable recommendations to enhance targeting, increase campaign effectiveness, and ultimately drive greater success for the financial institution (Moro et al., 2014).

Table C.3: Description of variables of the Bank Marketing dataset

Variable	Description	Type
x_1 = Age	Age of the client	Numerical discrete
x_2 = Job	Type of job	Categorical
x_3 = Marital	Marital status	Categorical
x_4 = Education	Level of education	Categorical
x_5 = Default	Has credit by default?	Categorical
x_6 = Balance	Account balance	Numerical discrete
x_7 = Housing	Has housing loan?	Categorical
x_8 = Loan	Has personal loan?	Categorical
x_9 = Contact	Type of communication contact	Categorical
x_{10} = Day	Last contact day of the month	Numerical discrete
x_{11} = Month	Last contact month of the year	Categorical
x_{12} = Campaign	Number of contacts performed during this campaign	Numerical discrete
x_{13} = Pdays	Days since the client was last contacted from a previous campaign	Numerical discrete
x_{14} = Previous	Number of contacts performed before this campaign	Numerical discrete
x_{15} = Poutcome	Outcome of the previous marketing campaign	Categorical
y = Subscribe	Client subscribed a term deposit	Categorical

C.1.3 Hotel Reservations dataset

The Hotel Reservations dataset, encompassing 36275 reservations, provides a snapshot of the dynamic online booking landscape, where cancellations and no-shows pose significant challenges for the industry. As detailed in Table C.4, the data captures various factors influencing customer behavior and booking outcomes. By predicting whether a customer will honor or cancel their reservation, models built on this dataset can empower hotels to proactively manage cancellations, optimize revenue strategies, and enhance operational efficiency. The dataset has no missing values.

C.1.4 Flight Delays dataset

The Flight Delays dataset provides detailed arrival and delay data for U.S. airports and carriers from August 2013 to August 2023. It includes metrics such as arrival counts, delays, cancellations, diversions, and reasons for delays. This dataset is useful for analyzing carrier performance,

Table C.4: Description of variables of the Hotel Reservations dataset

Variable	Description	Type
x_2 = # adults	Number of adults	Numerical discrete
x_3 = # children	Number of children	Numerical discrete
x_4 = # weekend_nights	Number of weekend nights stayed/booked	Numerical discrete
x_5 = # week nights	Number of weeknights stayed/booked	Numerical discrete
x_6 = Type of meal plan	Type of meal plan booked	Categorical
x_7 = Required car parking space	Car parking space requirement	Numerical discrete
x_8 = Room type reserved	Type of room reserved	Categorical
x_9 = Lead time	Days between booking and arrival date	Numerical discrete
x_{10} = Arrival year	Year of arrival date	Numerical discrete
x_{11} = Arrival month	Month of arrival date	Numerical discrete
x_{12} = Arrival date	Day of the month	Numerical discrete
x_{13} = Market segment type	Market segment designation	Categorical
x_{14} = Repeated guest	Repeated guest	Numerical discrete
x_{15} = # previous cancellations	Number of previous cancellations	Numerical discrete
x_{16} = # previous bookings not canceled	Number of previous bookings not canceled	Numerical discrete
x_{17} = Avg price per room	Average price per room per day	Numerical continuous
x_{18} = # special requests	Number of special requests	Numerical discrete
y = Booking status	Canceled reservation	Categorical

identifying trends, and understanding the factors contributing to Flight Delays, thereby aiding researchers and aviation professionals in their work.

For this analysis, a refined dataset was derived from the original, focusing on predicting Flight Delays and excluding irrelevant or duplicated variables. This refined dataset, detailed in Table C.5, enables the stratification of models both by U.S. states and regions and contains information on 539583 flights with no missing values.

Table C.5: Description of variables of the Flight Delays dataset

Variable	Description	Type
x_1 = Airline	Airline company	Categorical
x_2 = From airport	Departure airport	Categorical
x_3 = From state	Departure state	Categorical
x_4 = To airport	Arrival airport	Categorical
x_5 = To state	Arrival state	Categorical
x_6 = Weekday	Day of the week	Categorical
x_7 = Time	Departure time	Numerical discrete
x_8 = Length	Flight duration in minutes	Numerical discrete
y = Delay	Flight delay status (delayed or not)	Categorical

C.1.5 University of Porto dataset

The University of Porto dataset is a private dataset and also an expanded and revised version from the one presented in Table 4.1. It was assembled to train predictive models to predict student

success/failure in individual courses. This dataset, described in Table C.6, encompasses socio-demographic information and prior academic performance data for students across all university courses from the academic years 2011/2012 to 2015/2016. Missing values were replaced with ‘unknown’ for categorical variables and with the mean for numerical variables.

Table C.6: Description of variables of the University of Porto dataset

Variable	Description	Type
x_1 = Age	Age of the student	Numerical discrete
x_2 = Sex	Gender of the student	Categorical
x_3 = Marital status	Marital status	Categorical
x_4 = Nationality	Nationality of the student	Categorical
x_5 = Displaced	Is the student displaced?	Categorical
x_6 = Scholarship	Does the student have a scholarship?	Categorical
x_7 = Special needs	Does the student have special needs?	Categorical
x_8 = Type of admission	Type of admission contest	Categorical
x_9 = Type of student	Type of student (regular/mobility)	Categorical
x_{10} = Type of dedication	Type of dedication (full-time/part-time)	Categorical
x_{11} = Debt situation	Debt situation of the student	Categorical
x_{12} = Status of student	Current status of the student	Categorical
x_{13} = Accumulated years	Accumulated years of study	Numerical discrete
x_{14} = Delayed courses	Number of delayed courses	Numerical discrete
x_{15} = On going average	Current ongoing average grade on approved courses	Numerical continuous
x_{16} = On going average all	Current ongoing average grade for all courses	Numerical continuous
x_{17} = First year average	Average grade in the first year on approved courses	Numerical continuous
x_{18} = First year average all	Average grade in the first year for all course	Numerical continuous
x_{19} = Highest grade	Highest grade achieved	Numerical continuous
x_{20} = Lowest grade	Lowest grade achieved	Numerical continuous
x_{21} = Admission grade	Grade at the time of admission	Numerical continuous
x_{22} = Admission phase	Phase of admission	Categorical
x_{23} = First prgm option	First program option selected?	Categorical
x_{24} = Delayed years	Number of delayed years	Numerical discrete
x_{25} = Course enrollments	Number of course enrollments	Numerical discrete
x_{26} = Delayed ECTS	Number of delayed ECTS credits	Numerical continuous
y = Approval	Approval status (failed or passed)	Categorical

C.2 Business units and operational models

Operational models were developed for each dataset by grouping examples into distinct business units and training individual models for units that met predefined criteria. Table C.7 summarizes the datasets, their respective business units, target variables, positive classes for evaluation, minimum dataset size (*mds*) per unit, minimum positive class representation (*mpcr*) thresholds, and the resulting number of operational models created. Business units failing to meet the minimum requirements were excluded. Decision trees were trained for operational models using data from the business units in each dataset and evaluated using the C5.0 algorithm implementation of the R

language (Kuhn and Quinlan, 2023). To ensure robust and reliable performance estimates, 10-fold cross-validation was applied, repeated three times. The models were selected based on their predictive performance, using the Kappa metric. The predictive performance operational models are presented in Figure 7.2.

Table C.7: Operational model development summary

Dataset	Business unit	Target variable	Positive class	<i>mds</i>	<i>mpcr</i>	Op. models
Adult Census	Country of origin	Income level	$\geq 50K$	50	5%	42
Bank Marketing	Month	Subscribed campaign	yes	100	5%	12
Hotel Reservations	Arrival Year/Month	Booking status	cancelled	100	20%	18
Flight Delays	Departure airport	Delay	yes	100	10%	225
University of Porto	Course	Approval status	fail	100	10%	3073

C.2.1 Examples of operational models

The following are examples of operational models trained on each dataset:

Listing C.1: Operational model for country "France" (Adult Census dataset)

```

marital_status in {Divorced,Married-AF-spouse,Married-spouse-absent,
:
:         Never-married,Separated,Widowed}: <50K (649/36)
marital_status = Married-civ-spouse:
:...workclass = Never-worked: >=50K (0)
:   workclass in {Self-emp-inc,Self-emp-not-inc,State-gov,Unknown}:
:...relationship in {Other-relative,Unmarried}: <50K (8/4)
:   relationship in {Own-child,Wife}: >=50K (7/2)
:   relationship = Husband:
:   :...hours_per_week <= 21: <50K (3)
:   :   hours_per_week > 21: >=50K (54/4)
:   relationship = Not-in-family:
:   :...capital_gain <= -21: >=50K (3)
:   :   capital_gain > -21: <50K (4)
workclass in {Federal-gov,Local-gov,Private}:
:...hours_per_week <= 43: <50K (164/55)
:   hours_per_week > 43:
:   :...relationship in {Not-in-family,Other-relative,
:   :         Wife}: <50K (11/3)
:   :   relationship in {Own-child,Unmarried}: >=50K (1)
:   :   relationship = Husband:
:   :   :...age <= 37: <50K (6/1)
:   :   :   age > 37: >=50K (47/14)

```

C.3 Strata and consensus models

Consensus models were developed from groups (strata) of the operational models. The stratification is based on a unifying business concept, meaning that multiple operational models share

Listing C.2: Operational model for month "August" (Bank Marketing dataset)

```
previous <= 0: no (5142/403)
previous > 0:
:...poutcome = unknown: yes (1)
  poutcome in {failure,other}:
:...pdays <= 386: no (250/70)
  : pdays > 386: yes (36/15)
  poutcome = success:
:...pdays > 147: yes (71/14)
    pdays <= 147:
      ...contact in {telephone,unknown}: no (3)
        contact = cellular:
          ...pdays <= 101: yes (75/21)
            pdays > 101: no (44/13)
```

Listing C.3: Operational model for month "11/2017" (Hotel Reservations dataset)

```
lead_time <= 80: Not_Canceled (544/15)
lead_time > 80:
:...no_of_special_requests > 1: Not_Canceled (16)
  no_of_special_requests <= 1:
    ...market_segment_type in {Complementary,Corporate,
      :                               Offline}: Not_Canceled (6)
    market_segment_type = Online:
      ...arrival_date <= 25: Not_Canceled (11/4)
        arrival_date > 25: Canceled (5)
```

Listing C.4: Operational model for airport "ABE" (Flight Delays dataset)

```
time <= 731: no (164/32)
time > 731:
:...to_state in {FL,GA,NC}: no (30/8)
  to_state = IL:
    ...airline in {9E,EV,FL,US}: no (0)
    :   airline = XE: yes (12/5)
    :   airline = MQ:
    :     ...length <= 145: no (28/7)
    :     length > 145: yes (22/9)
  to_state = MI:
    ...weedday = friday: no (3)
      weedday in {monday,saturday,sunday,thursday,tuesday}: yes
        (39/12)
      weedday = wednesday:
        ...time <= 840: no (5/1)
          time > 840: yes (5/1)
```

Listing C.5: Operational model for course "18071" (University of Porto dataset)

```

delayed_courses <= 0: pass (92/16)
delayed_courses > 0:
:...admission_phase = phase 2: fail (3)
  admission_phase = unknown: pass (1)
  admission_phase = phase 1:
:...marital_status = divorced: fail (1)
  marital_status = married: pass (6/2)
  marital_status = single:
:...type_of_admission in {degree holder,none,other,over 23,
                           readmission,
                           :
                           special}: fail (26/6)
type_of_admission = general:
:...displaced = no:
:  :...delayed_courses <= 6: pass (35/15)
:  :  delayed_courses > 6: fail (6)
:  displaced = yes:
:  :...delayed_ects <= 18: fail (5)
:  :  delayed_ects > 18: pass (4/1)
type_of_admission = transfer:
:...status_of_student in {employed,other}: pass (1)
  status_of_student = ordinary:
:...delayed_ects <= 39: pass (9/2)
  :  delayed_ects > 39: fail (6/1)

```

the same concept. Except for the Adult Census dataset, all other datasets were experimented with two methods of stratifying the models. Table C.8 summarizes the datasets, identifying again the business units, stratification criteria, and the number of strata in each setting, which correspond to the subsequently developed consensus models.

Table C.8: Consensus model development summary

Dataset	Business unit	Stratification criteria	Consensus models
Adult Census	Country of origin	World regions	6
Bank Marketing	Month	Quarters	4
		Semesters	2
Hotel Reservations	Arrival Year/Month	Quarters	6
		Semesters	3
Flight Delays	Departure airport	US states and territories	55
		US regions	7
University of Porto	Course	Faculties	14
		Scientific areas by academic year	40

In the Bank Marketing dataset, the examples cover one year. The Hotel Reservations dataset, however, spans eighteen months, encompassing six quarters and three semesters. The Flight Delays dataset includes data from all fifty states added by five overseas territories (American Samoa, Guam, Northern Mariana Islands, Puerto Rico, and the U.S. Virgin Islands).

The following sections present the evaluation results using various metrics, consistently presented in two formats:

1. Models aggregated by strata – Each boxplot displays the distribution of multiple versions of the same model across the 10 setups. This analysis examines the robustness of the aggregation method against slight variations in the underlying datasets.
2. Models aggregated by setups – Each boxplot represents the distribution of all models within a specific setup. This analysis helps to evaluate the consistency of results across different setups and identify any potential biases.

Following the operational model creation, the *Inmplode* framework was employed to develop consensus models for each stratum. The merging function used for this process was an intersection, and the evaluation was conducted across 10 setups. The resulting consensus models are presented in four formats:

1. Operational tree models – These decision trees represent the original operational models trained from each business unit dataset.
2. Merged decision tables – These tables contain the rule sets obtained after merging the operational models using the intersection function. It is important to note that these tables are often incomplete before conversion to decision trees.
3. Synthetic data trees – These decision trees were trained on single-valued synthetic data (details provided in Section 6.2). They represent the consensus models derived from the synthetic data approach.
4. Distributional data trees – These decision trees were obtained from multi-valued distributional data (details provided in Section 6.3). They represent the consensus models derived from the symbolic data approach.

C.3.1 Examples of consensus models

The following are examples of consensus models developed within the context of each dataset.

C.4 Detailed results of evaluation

C.4.1 Results of evaluation of Predictive performance

The following tables present the average predictive performance of consensus models across the five datasets, followed by boxplots of the *Matthews Correlation Coefficient* (MCC) aggregated by strata and by setup. Although not a predictive performance metric, the performance results also include the *Fraction Covered* (FracCovered) (Equation 3.21) metric for the test data, which is calculated for the test set rather than the training set, where it is typically applied. This metric

Listing C.6: Consensus model for world region "South America" (Adult Census dataset)

```
capital_loss <= 1913:
:...sex = Female: <50K
:...sex = Male:
:   :...hours_per_week <= 42: <50K
:   :...hours_per_week > 42: >=50K
capital_loss > 1913:
:...age <= 38: >=50K
:...age > 38:
:   :...race in {Asian-Pac-Islander, Other}: >=50K
:   :...race in {Amer-Indian-Eskimo, Black, White}:
:     :...occupation in {Craft-repair, Farming-fishing, Machine-op-
:       inspct, Protective-serv, Unknown}: >=50K
:     :...occupation in {Adm-clerical, Armed-Forces, Exec-managerial,
:       Handlers-cleaners, Other-service, Priv-house-serv, Prof-
:       specialty, Sales, Tech-support, Transport-moving}: <50K
```

Listing C.7: Consensus model for semester "1st semester" (Bank Marketing dataset)

```
marital = single: yes
marital in {divorced, married}:
:...housing = yes: yes
:...housing = no:
:   :...job = retired: yes
:   :...job in {admin., blue-collar, entrepreneur, housemaid, management
:     , self-employed, services, student, technician, unemployed,
:     unknown}:
:   :...education = primary: yes
:   :...education in {secondary, tertiary, unknown}: no
```

Listing C.8: Consensus model for semeste "2017-2nd semester" (Hotel Reservations dataset)

```
avg_price_per_room <= 129:
:...room_type_reserved = Room_Type 2: Canceled
:...room_type_reserved in {Room_Type 1, Room_Type 3, Room_Type 4,
:   Room_Type 5, Room_Type 6, Room_Type 7}: Not_Canceled
avg_price_per_room > 129:
:...avg_price_per_room <= 151: Not_Canceled
:...avg_price_per_room > 151: Canceled
```

Listing C.9: Consensus model for US region "Pacific" (Flight Delays dataset)

```
to_state in {MN, NJ, OR, TX}: yes
to_state in {AK, AL, AR, AZ, CA, CO, CT, FL, GA, GU, HI, IA, ID, IL, IN,
:   KS, KY, LA, MA, MD, ME, MI, MO, MS, MT, NA, NC, ND, NE, NH, NM, NV,
:   NY, OH, OK, PA, PR, RI, SC, SD, TN, UT, VA, VI, VT, WA, WI, WV, WY
:   }:
:...time <= 1300: no
:...time > 1300:
:   :...airline in {AA, AS, DL, HA, UA, US}: no
:   :...airline in {9E, B6, CO, EV, F9, FL, MQ, OH, OO, WN, XE, YV}: yes
```

Listing C.10: Consensus model for faculty "Sports" (University of Porto dataset)

```

delayed_courses <= 0: pass
delayed_courses > 0:
...first_prgm_option = yes:
:   ...admission_grade <= 168: fail
:   ...admission_grade > 168: pass
...first_prgm_option in {no, unknown}: fail

```

measures the proportion of test set examples for which the a model provides predictions. While it is primarily related to completeness, it is important to present this metric within the context of predictive performance. Ranging from 0 to 1, where 1 represents full coverage (all test set examples have a prediction) and 0 indicates no coverage (none have an associated prediction).

The inclusion of FracCovered is due to the fact that some consensus models exhibit a zero coverage of the decision space (FracCovered = 0 as, for example, the consensus models as merged table representation of the ‘Midwest’, ‘Northeast’, ‘Southeast’, and ‘West’ in the Flight Delays dataset). In such cases, it becomes impossible to calculate the performance metrics outlined earlier. This limitation occurs because these metrics are defined as ratios involving the four quadrants of the confusion matrix (TP , FP , TN , FN). When the denominator in these ratios is zero, as happens when FracCovered = 0, the performance metrics are undefined and are thus denoted as ‘n/a’. Therefore, including the FracCovered metric in this analysis offers the necessary context to justify the absence of defined performance values in these particular instances.

Table C.9: Predictive performance of consensus models from the Adult Census dataset

Consensus model	Representation	FracCovered	Accuracy	F1	MCC
Asia	Merged table	0.466±0.093	0.945±0.037	0.006±0.019	0.021±0.000
	Synthetic data tree	0.693±0.206	0.691±0.168	0.202±0.182	0.082±0.069
	Distributional data tree	1.000±0.000	0.604±0.194	0.153±0.156	0.009±0.076
Central America	Merged table	0.461±0.046	0.953±0.008	0.023±0.046	0.037±0.031
	Synthetic data tree	0.717±0.285	0.689±0.152	0.252±0.194	0.121±0.135
	Distributional data tree	1.000±0.000	0.524±0.175	0.346±0.176	0.137±0.129
Europe	Merged table	0.316±0.208	0.955±0.007	0.106±0.175	0.469±0.092
	Synthetic data tree	0.768±0.243	0.648±0.117	0.510±0.031	0.294±0.056
	Distributional data tree	1.000±0.000	0.412±0.104	0.480±0.049	0.124±0.149
North America	Merged table	0.723±0.082	0.933±0.021	0.498±0.254	0.481±0.228
	Synthetic data tree	0.850±0.089	0.644±0.125	0.522±0.134	0.335±0.193
	Distributional data tree	1.000±0.000	0.615±0.205	0.378±0.212	0.209±0.248
South America	Merged table	0.781±0.148	0.898±0.045	0.354±0.217	0.366±0.160
	Synthetic data tree	0.719±0.267	0.510±0.210	0.366±0.096	0.201±0.116
	Distributional data tree	1.000±0.000	0.588±0.217	0.375±0.073	0.195±0.147
Unknown	Merged table	0.796±0.117	0.867±0.032	0.735±0.034	0.629±0.038
	Synthetic data tree	0.711±0.142	0.700±0.077	0.551±0.102	0.331±0.135
	Distributional data tree	1.000±0.000	0.594±0.172	0.432±0.171	0.201±0.230

Table C.10: Predictive performance of consensus models from the Bank Marketing dataset

Consensus model	Representation	FracCovered	Accuracy	F1	MCC
1st semester	Merged table	0.628±0.251	0.696±0.136	0.156±0.114	0.054±0.055
	Synthetic data tree	0.906±0.078	0.553±0.208	0.159±0.119	-0.041±0.068
	Distributional data tree	0.981±0.043	0.546±0.225	0.143±0.134	-0.036±0.067
2nd semester	Merged table	0.326±0.238	0.617±0.112	0.267±0.178	0.062±0.194
	Synthetic data tree	0.880±0.192	0.560±0.174	0.299±0.171	0.134±0.111
	Distributional data tree	1.000±0.000	0.473±0.190	0.405±0.019	0.135±0.133

Table C.11: Predictive performance of consensus models from the Hotel Reservations dataset

Consensus model	Representation	FracCovered	Accuracy	F1	MCC
2017-2nd semester	Merged table	0.247±0.147	0.880±0.105	0.166±0.209	0.092±0.081
	Synthetic data tree	0.968±0.031	0.544±0.058	0.255±0.048	0.083±0.073
	Distributional data tree	1.000±0.000	0.503±0.243	0.200±0.142	0.100±0.000
2018-1st semester	Merged table	0.050±0.057	0.638±0.121	0.401±0.316	0.117±0.120
	Synthetic data tree	0.978±0.034	0.574±0.114	0.309±0.136	0.070±0.086
	Distributional data tree	1.000±0.000	0.402±0.164	0.341±0.094	-0.021±0.068
2018-2nd semester	Merged table	0.189±0.251	0.724±0.107	0.318±0.315	0.229±0.191
	Synthetic data tree	0.987±0.021	0.478±0.071	0.549±0.018	0.126±0.059
	Distributional data tree	1.000±0.000	0.515±0.101	0.272±0.248	0.025±0.000

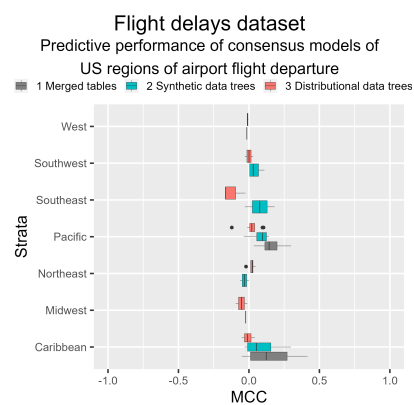
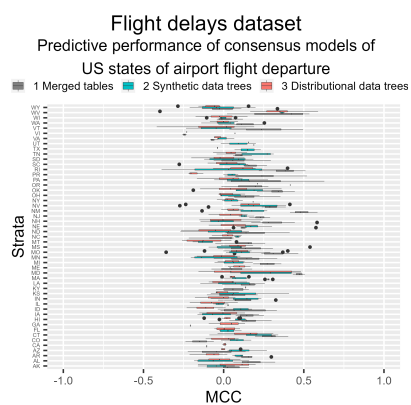
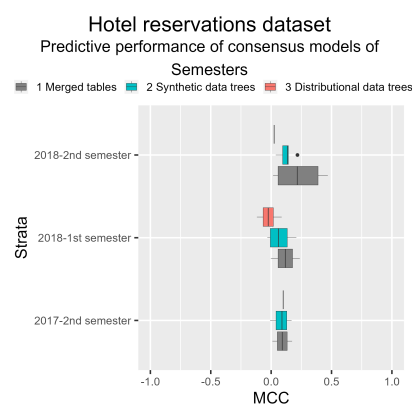
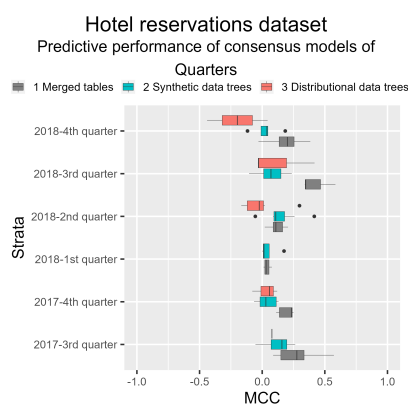
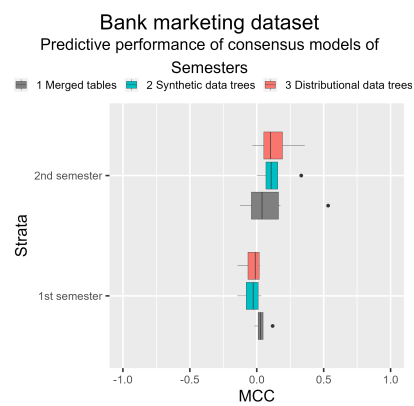
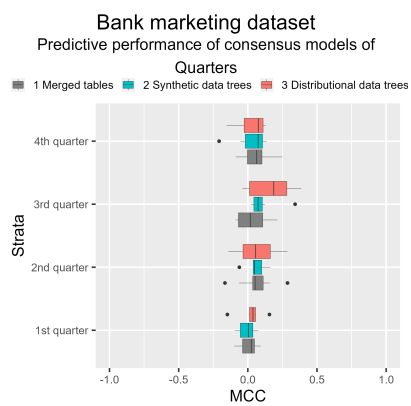
Table C.12: Predictive performance of consensus models from the Flight Delays dataset

Consensus model	Representation	FracCovered	Accuracy	F1	MCC
Caribbean	Merged table	0.921±0.129	0.609±0.083	0.406±0.128	0.155±0.154
	Synthetic data tree	0.671±0.275	0.505±0.092	0.401±0.135	0.093±0.129
	Distributional data tree	1.000±0.000	0.479±0.104	0.318±0.205	-0.007±0.038
Midwest	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.693±0.224	0.607±0.093	0.171±0.172	-0.023±0.000
	Distributional data tree	1.000±0.000	0.590±0.016	0.094±0.086	-0.053±0.042
Northeast	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.630±0.386	0.511±0.065	0.277±0.221	-0.032±0.031
	Distributional data tree	1.000±0.000	0.520±0.032	0.355±0.113	0.018±0.022
Pacific	Merged table	0.775±0.203	0.732±0.029	0.245±0.078	0.147±0.074
	Synthetic data tree	0.507±0.321	0.484±0.165	0.358±0.063	0.079±0.061
	Distributional data tree	1.000±0.000	0.682±0.071	0.137±0.061	0.020±0.062
Southeast	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.823±0.241	0.577±0.057	0.264±0.218	0.077±0.105
	Distributional data tree	1.000±0.000	0.465±0.101	0.295±0.143	-0.119±0.067
Southwest	Merged table	0.014±0.021	0.648±0.241	0.000±0.000	n/a
	Synthetic data tree	0.657±0.197	0.617±0.071	0.228±0.099	0.043±0.043
	Distributional data tree	1.000±0.000	0.524±0.096	0.286±0.203	0.001±0.029
West	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.635±0.365	0.454±0.048	0.474±0.096	-0.015±0.000
	Distributional data tree	1.000±0.000	0.527±0.046	0.314±0.137	-0.010±0.005

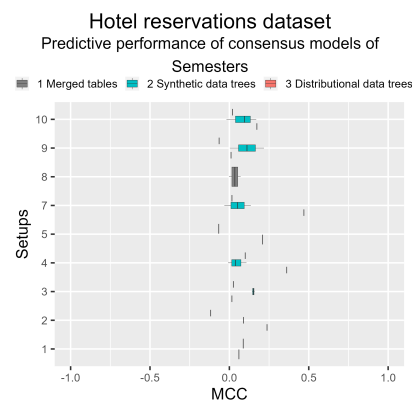
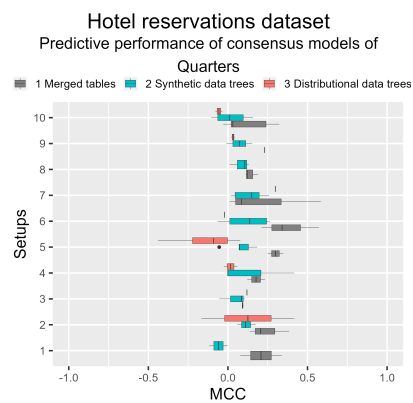
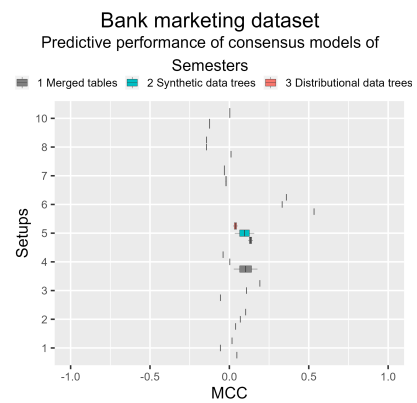
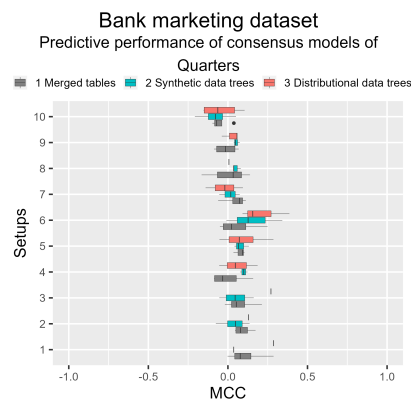
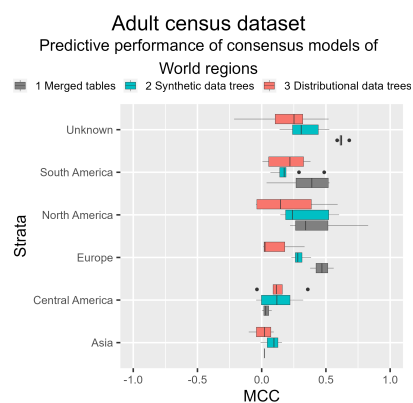
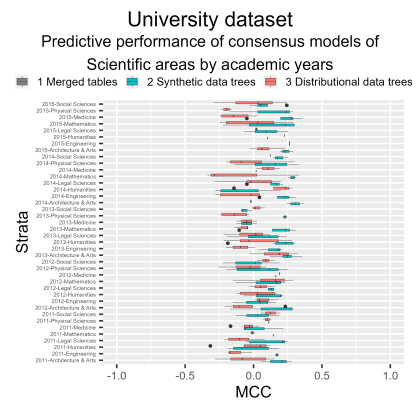
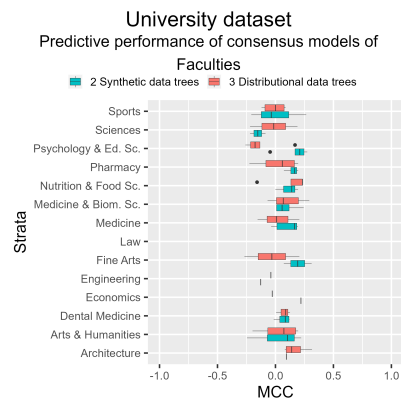
Table C.13: Predictive performance of consensus models from the University of Porto dataset

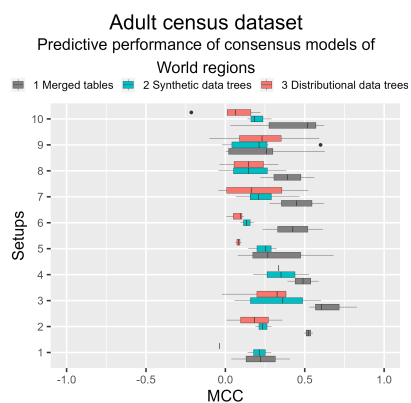
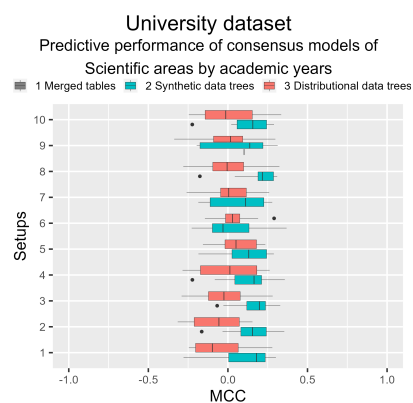
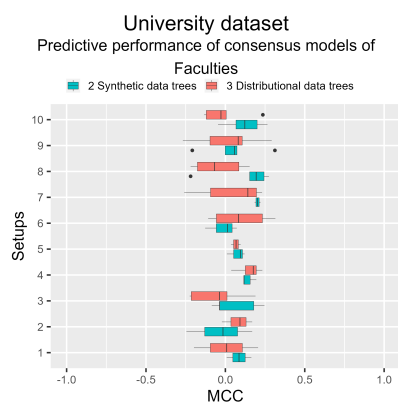
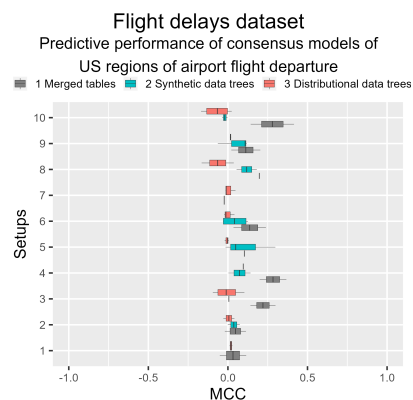
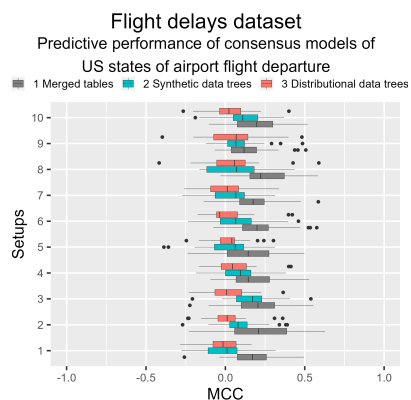
Consensus model	Representation	FracCovered	Accuracy	F1	MCC
Architecture	Merged table	0.004±0.010	n/a	n/a	n/a
	Synthetic data tree	0.992±0.014	0.653±0.211	0.138±0.118	0.094±0.000
	Distributional data tree	1.000±0.000	0.624±0.207	0.232±0.079	0.169±0.093
Arts & Humanities	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.957±0.074	0.483±0.152	0.402±0.088	0.027±0.198
	Distributional data tree	1.000±0.000	0.499±0.137	0.398±0.038	0.036±0.159
Dental Medicine	Merged table	0.020±0.028	0.444±0.152	0.466±0.157	n/a
	Synthetic data tree	1.000±0.000	0.512±0.036	0.542±0.068	0.069±0.055
	Distributional data tree	1.000±0.000	0.518±0.025	0.448±0.076	0.076±0.042
Economics	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.589±0.000	0.487±0.000	0.220±0.000
	Distributional data tree	1.000±0.000	0.553±0.000	0.209±0.000	-0.028±0.000
Engineering	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.505±0.139	0.218±0.217	-0.127±0.000
	Distributional data tree	1.000±0.000	0.535±0.109	0.218±0.217	-0.040±0.000
Fine Arts	Merged table	0.055±0.070	0.906±0.026	n/a	n/a
	Synthetic data tree	1.000±0.000	0.508±0.163	0.374±0.128	0.192±0.120
	Distributional data tree	1.000±0.000	0.518±0.166	0.309±0.063	-0.031±0.237
Law	Merged table	0.001±0.003	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.650±0.000	0.083±0.000	n/a
	Distributional data tree	n/a	n/a	n/a	n/a
Medicine	Merged table	0.039±0.041	0.940±0.000	n/a	n/a
	Synthetic data tree	0.999±0.002	0.666±0.166	0.207±0.065	0.104±0.096
	Distributional data tree	1.000±0.000	0.627±0.116	0.189±0.089	0.021±0.147
Medic./Biom. Sc.	Merged table	0.003±0.008	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.529±0.142	0.347±0.124	0.087±0.088
	Distributional data tree	1.000±0.000	0.632±0.087	0.269±0.130	0.099±0.130
Nutrition/Food Sc.	Merged table	0.115±0.062	0.732±0.179	n/a	n/a
	Synthetic data tree	0.942±0.098	0.526±0.168	0.287±0.072	0.111±0.083
	Distributional data tree	1.000±0.000	0.678±0.072	0.249±0.103	0.135±0.170
Pharmacy	Merged table	0.006±0.018	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.561±0.147	0.329±0.090	0.150±0.047
	Distributional data tree	1.000±0.000	0.535±0.134	0.329±0.077	0.022±0.167
Psychology/Ed. Sc.	Merged table	0.037±0.042	0.913±0.030	n/a	n/a
	Synthetic data tree	0.987±0.026	0.672±0.110	0.259±0.131	0.171±0.114
	Distributional data tree	1.000±0.000	0.468±0.215	0.196±0.082	-0.123±0.152
Sciences	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.419±0.019	0.533±0.019	-0.152±0.067
	Distributional data tree	1.000±0.000	0.496±0.096	0.509±0.004	-0.015±0.204
Sports	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.504±0.225	0.243±0.149	0.007±0.196
	Distributional data tree	1.000±0.000	0.525±0.101	0.245±0.110	-0.009±0.085

Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results





C.4.2 Results of evaluation of Predictive performance loss

The following tables present the average predictive performance loss of consensus models across the five datasets, followed by boxplots of $\overline{\Delta\text{MCC}}$ (the difference of MCC of consensus and operational models) aggregated by strata and by setup.

Table C.14: Predictive performance loss of consensus models from the Adult Census dataset

Consensus model	Representation	FracCovered	$\overline{\Delta\text{Accuracy}}$	$\overline{\Delta\text{FI}}$	$\overline{\Delta\text{MCC}}$
Asia	Merged table	0.466±0.093	0.113±0.039	-0.645±0.031	-0.513±0.000
	Synthetic data tree	0.693±0.206	-0.132±0.164	-0.432±0.189	-0.437±0.079
	Distributional data tree	1.000±0.000	-0.219±0.194	-0.481±0.162	-0.514±0.081
Central America	Merged table	0.461±0.046	0.121±0.009	-0.676±0.050	-0.554±0.034
	Synthetic data tree	0.717±0.285	-0.143±0.150	-0.447±0.197	-0.462±0.139
	Distributional data tree	1.000±0.000	-0.310±0.178	-0.356±0.175	-0.443±0.134
Europe	Merged table	0.316±0.208	0.134±0.014	-0.568±0.166	-0.096±0.085
	Synthetic data tree	0.768±0.243	-0.178±0.116	-0.181±0.038	-0.275±0.053
	Distributional data tree	1.000±0.000	-0.415±0.102	-0.212±0.049	-0.449±0.150
North America	Merged table	0.723±0.082	0.055±0.023	-0.200±0.247	-0.139±0.220
	Synthetic data tree	0.850±0.089	-0.233±0.122	-0.172±0.123	-0.281±0.179
	Distributional data tree	1.000±0.000	-0.260±0.203	-0.311±0.193	-0.401±0.235
South America	Merged table	0.781±0.148	0.048±0.043	-0.251±0.233	-0.152±0.170
	Synthetic data tree	0.719±0.267	-0.337±0.217	-0.237±0.086	-0.317±0.117
	Distributional data tree	1.000±0.000	-0.258±0.217	-0.225±0.087	-0.318±0.157
Unknown	Merged table	0.796±0.117	0.035±0.031	0.017±0.045	0.047±0.048
	Synthetic data tree	0.711±0.142	-0.130±0.090	-0.167±0.102	-0.249±0.148
	Distributional data tree	1.000±0.000	-0.235±0.178	-0.289±0.166	-0.378±0.242

Table C.15: Predictive performance loss of consensus models from the Bank Marketing dataset

Consensus model	Representation	FracCovered	$\overline{\Delta\text{Accuracy}}$	$\overline{\Delta\text{FI}}$	$\overline{\Delta\text{MCC}}$
1st semester	Merged table	0.628±0.251	-0.115±0.145	-0.175±0.113	-0.204±0.067
	Synthetic data tree	0.906±0.078	-0.268±0.209	-0.185±0.139	-0.305±0.065
	Distributional data tree	0.981±0.043	-0.272±0.230	-0.193±0.143	-0.299±0.061
2nd semester	Merged table	0.326±0.238	-0.120±0.107	-0.078±0.185	-0.131±0.188
	Synthetic data tree	0.880±0.192	-0.184±0.173	-0.053±0.185	-0.065±0.071
	Distributional data tree	1.000±0.000	-0.272±0.177	0.054±0.030	-0.070±0.098

Table C.16: Predictive performance loss of consensus models from the Hotel Reservations dataset

Consensus model	Representation	FracCovered	$\overline{\Delta\text{Accuracy}}$	$\overline{\Delta\text{F1}}$	$\overline{\Delta\text{MCC}}$
2017-2nd semester	Merged table	0.247 $\pm$ 0.147	-0.009 $\pm$ 0.108	-0.166 $\pm$ 0.204	-0.306 $\pm$ 0.014
	Synthetic data tree	0.968 $\pm$ 0.031	-0.371 $\pm$ 0.052	-0.125 $\pm$ 0.107	-0.480 $\pm$ 0.136
	Distributional data tree	1.000 $\pm$ 0.000	-0.413 $\pm$ 0.258	-0.181 $\pm$ 0.092	-0.546 $\pm$ 0.000
2018-1st semester	Merged table	0.050 $\pm$ 0.057	-0.113 $\pm$ 0.125	0.219 $\pm$ 0.313	-0.188 $\pm$ 0.078
	Synthetic data tree	0.978 $\pm$ 0.034	-0.181 $\pm$ 0.115	0.126 $\pm$ 0.142	-0.309 $\pm$ 0.136
	Distributional data tree	1.000 $\pm$ 0.000	-0.353 $\pm$ 0.171	0.158 $\pm$ 0.073	-0.403 $\pm$ 0.108
2018-2nd semester	Merged table	0.189 $\pm$ 0.251	-0.025 $\pm$ 0.127	-0.222 $\pm$ 0.340	-0.342 $\pm$ 0.180
	Synthetic data tree	0.987 $\pm$ 0.021	-0.268 $\pm$ 0.066	0.020 $\pm$ 0.116	-0.437 $\pm$ 0.059
	Distributional data tree	1.000 $\pm$ 0.000	-0.231 $\pm$ 0.123	-0.257 $\pm$ 0.183	-0.508 $\pm$ 0.000

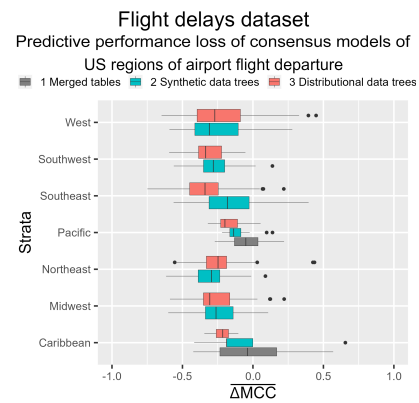
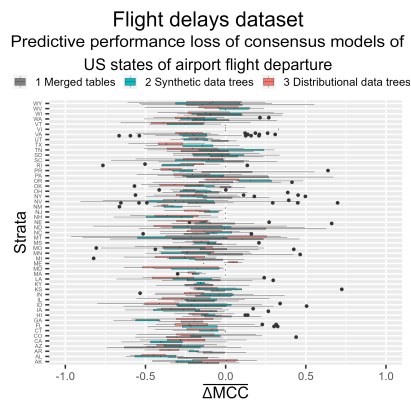
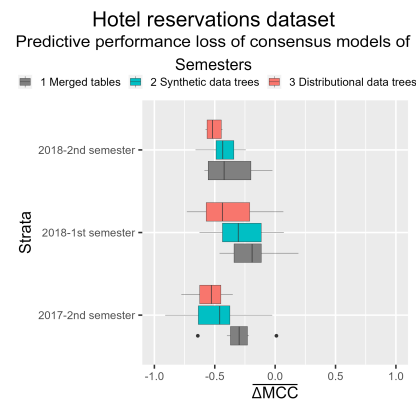
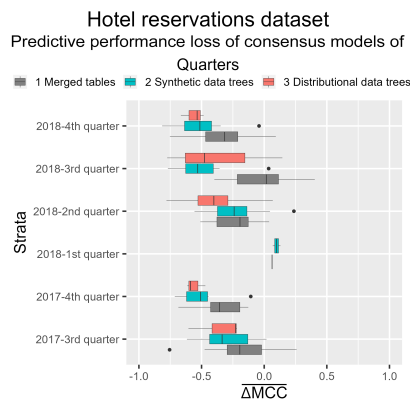
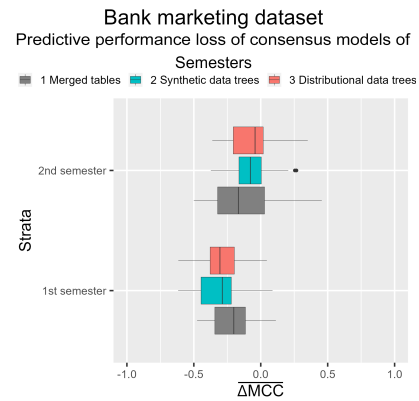
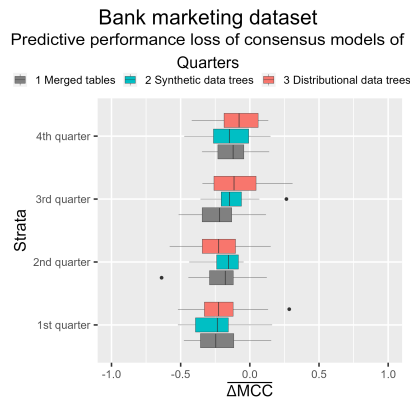
Table C.17: Predictive performance loss of consensus models from the Flight Delays dataset

Consensus model	Representation	FracCovered	$\overline{\Delta\text{Accuracy}}$	$\overline{\Delta\text{F1}}$	$\overline{\Delta\text{MCC}}$
Caribbean	Merged table	0.921 $\pm$ 0.129	-0.028 $\pm$ 0.073	0.196 $\pm$ 0.133	0.001 $\pm$ 0.142
	Synthetic data tree	0.671 $\pm$ 0.275	-0.135 $\pm$ 0.112	0.181 $\pm$ 0.100	-0.093 $\pm$ 0.240
	Distributional data tree	1.000 $\pm$ 0.000	-0.161 $\pm$ 0.124	0.109 $\pm$ 0.264	-0.236 $\pm$ 0.078
Midwest	Merged table	0.000 $\pm$ 0.000	n/a	n/a	n/a
	Synthetic data tree	0.693 $\pm$ 0.224	-0.046 $\pm$ 0.093	-0.144 $\pm$ 0.158	-0.238 $\pm$ 0.000
	Distributional data tree	1.000 $\pm$ 0.000	-0.063 $\pm$ 0.020	-0.221 $\pm$ 0.072	-0.261 $\pm$ 0.035
Northeast	Merged table	0.000 $\pm$ 0.000	n/a	n/a	n/a
	Synthetic data tree	0.630 $\pm$ 0.386	-0.143 $\pm$ 0.071	-0.200 $\pm$ 0.213	-0.307 $\pm$ 0.032
	Distributional data tree	1.000 $\pm$ 0.000	-0.136 $\pm$ 0.035	-0.120 $\pm$ 0.122	-0.249 $\pm$ 0.044
Pacific	Merged table	0.775 $\pm$ 0.203	-0.003 $\pm$ 0.026	0.109 $\pm$ 0.112	-0.037 $\pm$ 0.080
	Synthetic data tree	0.507 $\pm$ 0.321	-0.251 $\pm$ 0.164	0.221 $\pm$ 0.091	-0.109 $\pm$ 0.039
	Distributional data tree	1.000 $\pm$ 0.000	-0.053 $\pm$ 0.070	0.004 $\pm$ 0.089	-0.165 $\pm$ 0.054
Southeast	Merged table	0.000 $\pm$ 0.000	n/a	n/a	n/a
	Synthetic data tree	0.823 $\pm$ 0.241	-0.102 $\pm$ 0.066	-0.025 $\pm$ 0.208	-0.163 $\pm$ 0.112
	Distributional data tree	1.000 $\pm$ 0.000	-0.213 $\pm$ 0.093	0.012 $\pm$ 0.154	-0.342 $\pm$ 0.053
Southwest	Merged table	0.014 $\pm$ 0.021	-0.065 $\pm$ 0.242	-0.422 $\pm$ 0.000	n/a
	Synthetic data tree	0.657 $\pm$ 0.197	-0.098 $\pm$ 0.065	-0.207 $\pm$ 0.120	-0.267 $\pm$ 0.043
	Distributional data tree	1.000 $\pm$ 0.000	-0.188 $\pm$ 0.093	-0.142 $\pm$ 0.211	-0.319 $\pm$ 0.059
West	Merged table	0.000 $\pm$ 0.000	n/a	n/a	n/a
	Synthetic data tree	0.635 $\pm$ 0.365	-0.197 $\pm$ 0.041	0.104 $\pm$ 0.105	-0.247 $\pm$ 0.000
	Distributional data tree	1.000 $\pm$ 0.000	-0.124 $\pm$ 0.039	-0.056 $\pm$ 0.146	-0.225 $\pm$ 0.011

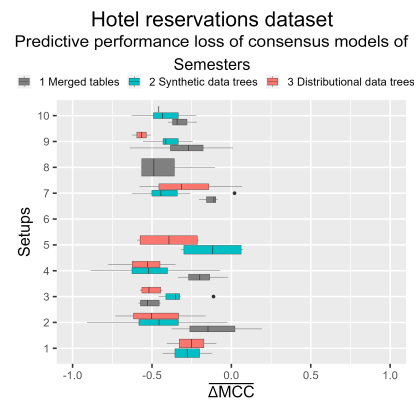
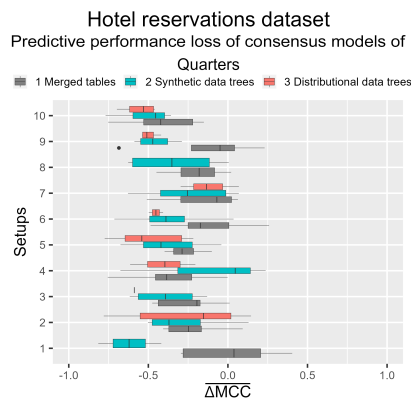
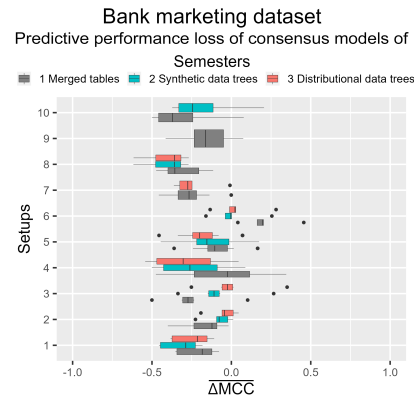
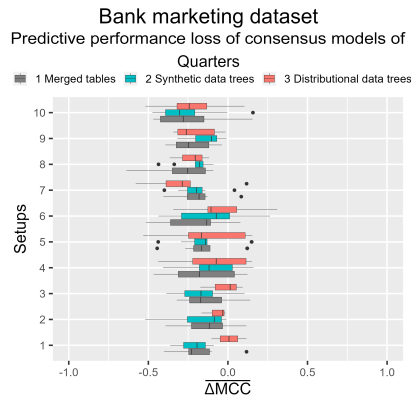
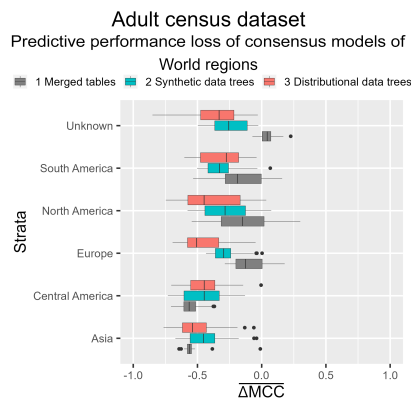
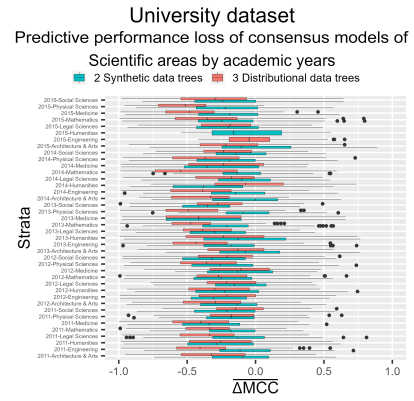
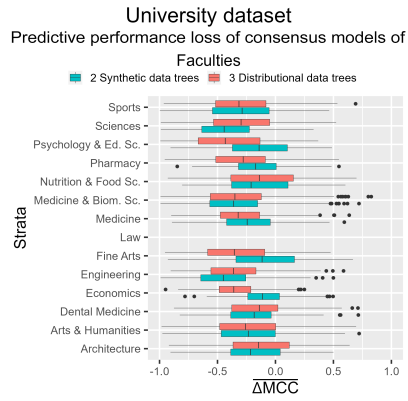
Table C.18: Predictive performance loss of consensus models from the University of Porto dataset

Consensus model	Representation	FracCovered	$\overline{\Delta\text{Accuracy}}$	$\overline{\Delta\text{F1}}$	$\overline{\Delta\text{MCC}}$
Architecture	Merged table	0.004±0.010	n/a	n/a	n/a
	Synthetic data tree	0.992±0.014	-0.129±0.217	-0.130±0.111	-0.194±0.000
	Distributional data tree	1.000±0.000	-0.155±0.214	-0.031±0.065	-0.134±0.093
Arts & Humanities	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.957±0.074	-0.224±0.152	0.037±0.088	-0.239±0.185
	Distributional data tree	1.000±0.000	-0.208±0.137	0.030±0.040	-0.241±0.156
Dental Medicine	Merged table	0.020±0.028	-0.225±0.166	-0.098±0.184	n/a
	Synthetic data tree	1.000±0.000	-0.150±0.037	0.002±0.084	-0.185±0.069
	Distributional data tree	1.000±0.000	-0.142±0.036	-0.096±0.092	-0.147±0.083
Economics	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.135±0.000	0.066±0.000	-0.098±0.000
	Distributional data tree	1.000±0.000	-0.171±0.000	-0.212±0.000	-0.345±0.000
Engineering	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.213±0.137	-0.240±0.226	-0.439±0.000
	Distributional data tree	1.000±0.000	-0.182±0.107	-0.240±0.226	-0.351±0.000
Fine Arts	Merged table	0.055±0.070	0.189±0.035	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.208±0.158	0.024±0.137	-0.111±0.128
	Distributional data tree	1.000±0.000	-0.201±0.167	-0.054±0.060	-0.334±0.229
Law	Merged table	0.001±0.003	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.070±0.000	-0.294±0.000	n/a
	Distributional data tree	n/a	n/a	n/a	n/a
Medicine	Merged table	0.039±0.041	0.145±0.000	n/a	n/a
	Synthetic data tree	0.999±0.002	-0.132±0.167	-0.085±0.063	-0.230±0.088
	Distributional data tree	1.000±0.000	-0.173±0.125	-0.097±0.083	-0.307±0.131
Medic./Biom. Sc.	Merged table	0.003±0.008	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.250±0.142	-0.116±0.118	-0.340±0.083
	Distributional data tree	1.000±0.000	-0.147±0.094	-0.194±0.137	-0.328±0.153
Nutrition/Food Sc.	Merged table	0.115±0.062	-0.032±0.175	n/a	n/a
	Synthetic data tree	0.942±0.098	-0.238±0.164	0.089±0.091	-0.162±0.113
	Distributional data tree	1.000±0.000	-0.085±0.070	0.048±0.117	-0.136±0.193
Pharmacy	Merged table	0.006±0.018	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.191±0.141	-0.041±0.108	-0.155±0.061
	Distributional data tree	1.000±0.000	-0.220±0.136	-0.045±0.095	-0.283±0.180
Psychology/Ed. Sc.	Merged table	0.037±0.042	0.136±0.034	n/a	n/a
	Synthetic data tree	0.987±0.026	-0.106±0.115	0.017±0.127	-0.129±0.102
	Distributional data tree	1.000±0.000	-0.310±0.220	-0.046±0.077	-0.423±0.159
Sciences	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.273±0.021	-0.009±0.021	-0.434±0.070
	Distributional data tree	1.000±0.000	-0.196±0.098	-0.032±0.002	-0.297±0.207
Sports	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	-0.268±0.225	-0.022±0.155	-0.292±0.193
	Distributional data tree	1.000±0.000	-0.247±0.103	-0.020±0.112	-0.313±0.094

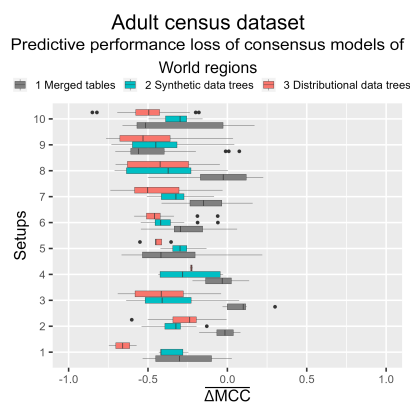
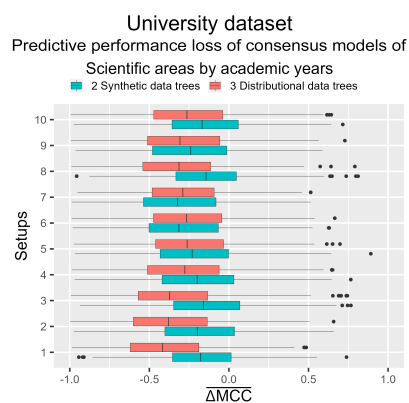
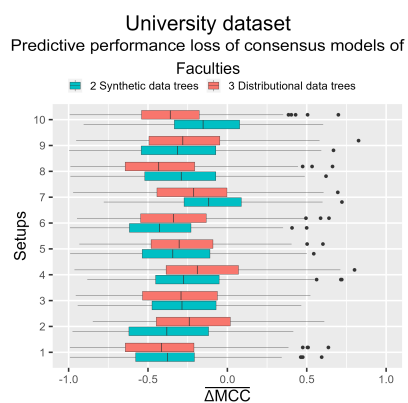
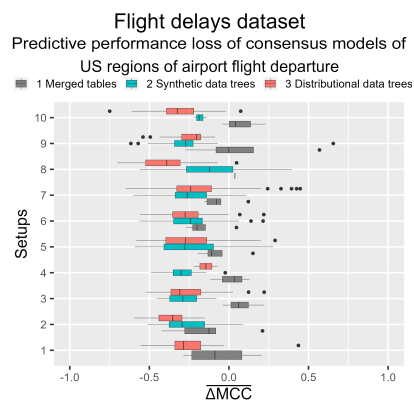
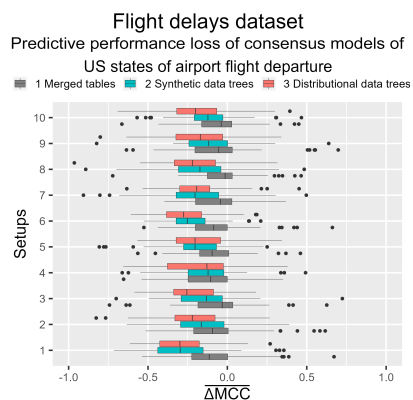
Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



C.4.3 Results of evaluation of Fidelity

The following tables present the average fidelity of consensus models across the five datasets, followed by boxplots of FidMCC (Fidelity calculated with MCC) aggregated by strata and by setup.

Table C.19: Fidelity of consensus models from the Adult Census dataset

Consensus model	Representation	FracCovered	FidAccuracy	FidF1	FidMCC
Asia	Merged table	0.466±0.093	0.985±0.041	0.024±0.000	n/a
	Synthetic data tree	0.693±0.206	0.697±0.185	0.189±0.177	0.135±0.036
	Distributional data tree	1.000±0.000	0.596±0.205	0.143±0.139	-0.008±0.086
Central America	Merged table	0.461±0.046	0.995±0.008	0.129±0.020	n/a
	Synthetic data tree	0.717±0.285	0.712±0.174	0.264±0.193	0.147±0.144
	Distributional data tree	1.000±0.000	0.530±0.193	0.342±0.163	0.162±0.129
Europe	Merged table	0.316±0.208	0.997±0.003	0.944±0.056	0.943±0.057
	Synthetic data tree	0.768±0.243	0.652±0.124	0.508±0.039	0.308±0.067
	Distributional data tree	1.000±0.000	0.414±0.113	0.473±0.058	0.140±0.166
North America	Merged table	0.723±0.082	0.986±0.012	0.702±0.209	0.905±0.067
	Synthetic data tree	0.850±0.089	0.654±0.134	0.518±0.152	0.360±0.213
	Distributional data tree	1.000±0.000	0.618±0.221	0.360±0.222	0.203±0.270
South America	Merged table	0.781±0.148	0.946±0.044	0.539±0.187	0.682±0.193
	Synthetic data tree	0.719±0.267	0.499±0.230	0.349±0.120	0.201±0.163
	Distributional data tree	1.000±0.000	0.588±0.215	0.359±0.075	0.195±0.146
Unknown	Merged table	0.796±0.117	0.956±0.025	0.887±0.047	0.865±0.054
	Synthetic data tree	0.711±0.142	0.699±0.085	0.547±0.130	0.344±0.177
	Distributional data tree	1.000±0.000	0.600±0.174	0.451±0.187	0.227±0.277

Table C.20: Fidelity of consensus models from the Bank Marketing dataset

Consensus model	Representation	FracCovered	FidAccuracy	FidF1	FidMCC
1st semester	Merged table	0.628±0.251	0.753±0.135	0.158±0.084	n/a
	Synthetic data tree	0.906±0.078	0.559±0.249	0.110±0.056	-0.083±0.090
	Distributional data tree	0.981±0.043	0.548±0.275	0.087±0.084	-0.071±0.099
2nd semester	Merged table	0.326±0.238	0.727±0.149	0.244±0.191	0.500±0.295
	Synthetic data tree	0.880±0.192	0.569±0.253	0.256±0.193	0.200±0.279
	Distributional data tree	1.000±0.000	0.476±0.278	0.387±0.126	0.278±0.258

Table C.21: Fidelity of consensus models from the Hotel Reservations dataset

Consensus model	Representation	FracCovered	FidAccuracy	FidF1	FidMCC
2017-2nd semester	Merged table	0.247±0.147	0.895±0.140	0.183±0.131	n/a
	Synthetic data tree	0.968±0.031	0.532±0.067	0.152±0.038	-0.017±0.048
	Distributional data tree	1.000±0.000	0.500±0.288	0.149±0.107	0.115±0.000
2018-1st semester	Merged table	0.050±0.057	0.551±0.400	0.000±0.000	n/a
	Synthetic data tree	0.978±0.034	0.594±0.214	0.101±0.072	0.112±0.187
	Distributional data tree	1.000±0.000	0.334±0.309	0.086±0.047	0.015±0.075
2018-2nd semester	Merged table	0.189±0.251	0.775±0.259	0.209±0.184	n/a
	Synthetic data tree	0.987±0.021	0.359±0.155	0.342±0.085	0.142±0.076
	Distributional data tree	1.000±0.000	0.584±0.264	0.188±0.187	-0.039±0.000

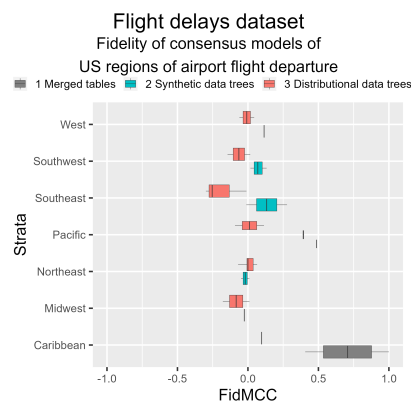
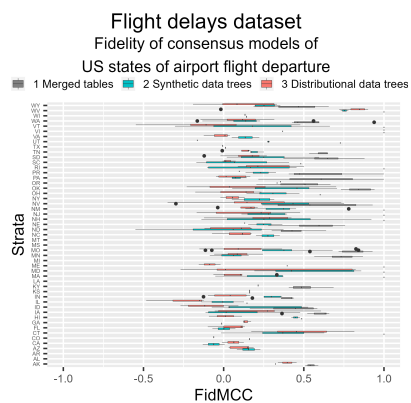
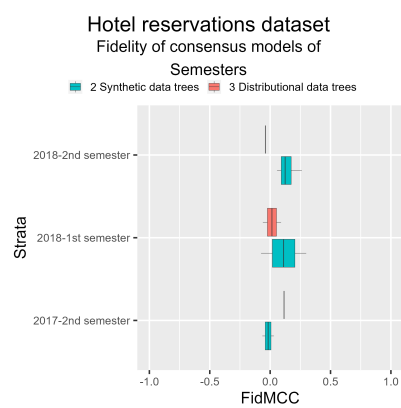
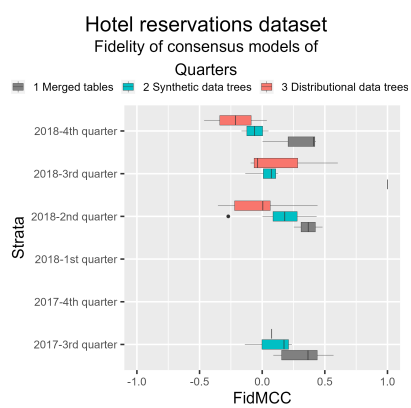
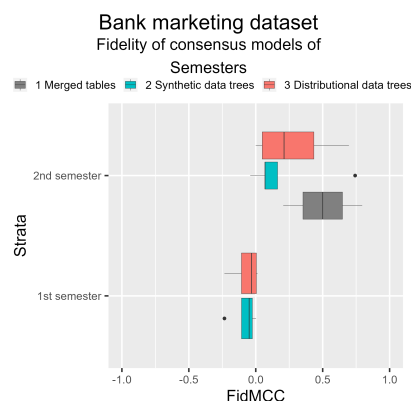
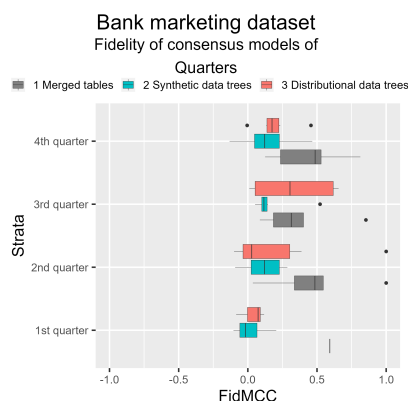
Table C.22: Fidelity of consensus models from the Flight Delays dataset

Consensus model	Representation	FracCovered	FidAccuracy	FidF1	FidMCC
Caribbean	Merged table	0.921±0.129	0.787±0.111	0.472±0.118	0.697±0.205
	Synthetic data tree	0.671±0.275	0.551±0.179	0.289±0.111	n/a
	Distributional data tree	1.000±0.000	0.441±0.285	0.129±0.085	0.097±0.000
Midwest	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.693±0.224	0.704±0.155	0.174±0.123	-0.026±0.000
	Distributional data tree	1.000±0.000	0.719±0.036	0.054±0.063	-0.084±0.094
Northeast	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.630±0.386	0.546±0.138	0.226±0.177	-0.019±0.030
	Distributional data tree	1.000±0.000	0.530±0.077	0.269±0.076	0.005±0.046
Pacific	Merged table	0.775±0.203	0.896±0.032	0.226±0.086	0.487±0.000
	Synthetic data tree	0.507±0.321	0.478±0.279	0.210±0.084	0.393±0.004
	Distributional data tree	1.000±0.000	0.840±0.111	0.057±0.046	0.011±0.103
Southeast	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.823±0.241	0.640±0.154	0.153±0.131	0.133±0.144
	Distributional data tree	1.000±0.000	0.470±0.217	0.129±0.085	-0.188±0.126
Southwest	Merged table	0.014±0.021	0.648±0.352	0.000±0.000	n/a
	Synthetic data tree	0.657±0.197	0.719±0.129	0.264±0.104	0.073±0.047
	Distributional data tree	1.000±0.000	0.540±0.174	0.192±0.160	-0.064±0.080
West	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.635±0.365	0.426±0.119	0.363±0.043	0.115±0.000
	Distributional data tree	1.000±0.000	0.573±0.106	0.238±0.082	-0.008±0.053

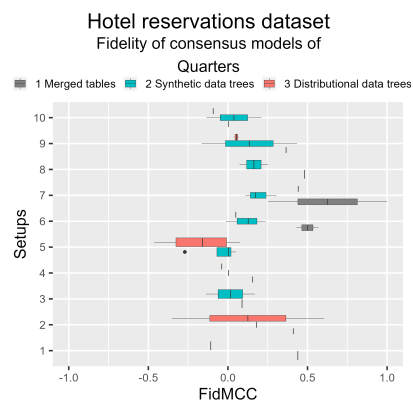
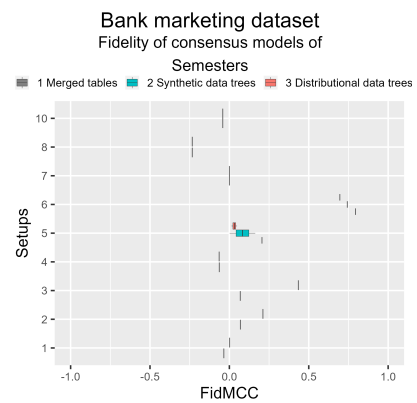
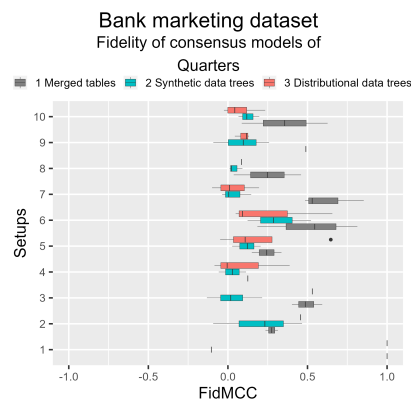
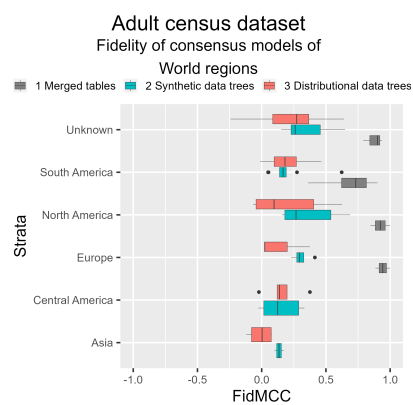
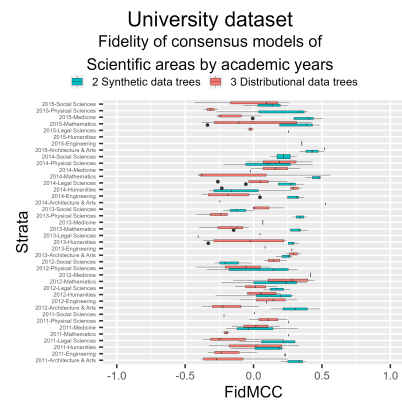
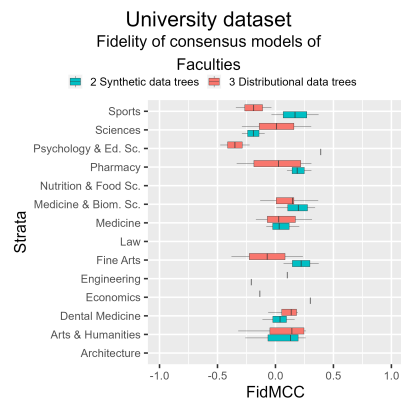
Table C.23: Fidelity of consensus models from the University of Porto dataset

Consensus model	Representation	FracCovered	FidAccuracy	FidF1	FidMCC
Architecture	Merged table	0.004±0.010	n/a	n/a	n/a
	Synthetic data tree	0.992±0.014	0.674±0.265	0.082±0.067	n/a
	Distributional data tree	1.000±0.000	0.629±0.257	0.139±0.067	n/a
Arts & Humanities	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	0.957±0.074	0.461±0.174	0.295±0.051	0.044±0.221
	Distributional data tree	1.000±0.000	0.487±0.158	0.296±0.035	0.056±0.234
Dental Medicine	Merged table	0.020±0.028	0.656±0.081	0.660±0.077	n/a
	Synthetic data tree	1.000±0.000	0.494±0.034	0.485±0.090	0.033±0.101
	Distributional data tree	1.000±0.000	0.501±0.031	0.415±0.059	0.104±0.092
Economics	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.588±0.000	0.428±0.000	0.300±0.000
	Distributional data tree	1.000±0.000	0.568±0.000	0.137±0.000	-0.134±0.000
Engineering	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.503±0.175	0.171±0.169	-0.209±0.000
	Distributional data tree	1.000±0.000	0.556±0.122	0.201±0.200	0.102±0.000
Fine Arts	Merged table	0.055±0.070	0.875±0.030	n/a	n/a
	Synthetic data tree	1.000±0.000	0.487±0.191	0.288±0.103	0.221±0.152
	Distributional data tree	1.000±0.000	0.512±0.177	0.235±0.074	-0.072±0.308
Law	Merged table	0.001±0.003	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.709±0.000	0.049±0.000	n/a
	Distributional data tree	n/a	n/a	n/a	n/a
Medicine	Merged table	0.039±0.041	0.945±0.000	n/a	n/a
	Synthetic data tree	0.999±0.002	0.676±0.185	0.146±0.029	0.053±0.117
	Distributional data tree	1.000±0.000	0.653±0.110	0.134±0.088	0.058±0.199
Medic./Biom. Sc.	Merged table	0.003±0.008	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.537±0.158	0.299±0.124	0.186±0.126
	Distributional data tree	1.000±0.000	0.639±0.085	0.224±0.114	0.110±0.167
Nutrition/Food Sc.	Merged table	0.115±0.062	0.804±0.230	n/a	n/a
	Synthetic data tree	0.942±0.098	0.530±0.231	0.174±0.042	n/a
	Distributional data tree	1.000±0.000	0.745±0.089	0.175±0.082	n/a
Pharmacy	Merged table	0.006±0.018	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.569±0.179	0.274±0.060	0.199±0.086
	Distributional data tree	1.000±0.000	0.537±0.153	0.254±0.074	0.007±0.256
Psychology/Ed. Sc.	Merged table	0.037±0.042	0.949±0.051	n/a	n/a
	Synthetic data tree	0.987±0.026	0.699±0.146	0.171±0.100	0.391±0.000
	Distributional data tree	1.000±0.000	0.468±0.261	0.085±0.050	-0.350±0.127
Sciences	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.411±0.028	0.476±0.028	-0.191±0.098
	Distributional data tree	1.000±0.000	0.498±0.115	0.476±0.027	0.010±0.299
Sports	Merged table	0.000±0.000	n/a	n/a	n/a
	Synthetic data tree	1.000±0.000	0.505±0.307	0.152±0.095	0.169±0.203
	Distributional data tree	1.000±0.000	0.539±0.135	0.160±0.099	-0.188±0.153

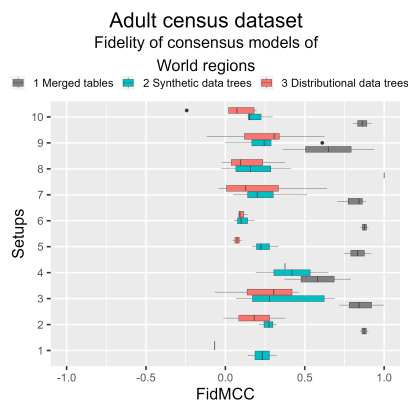
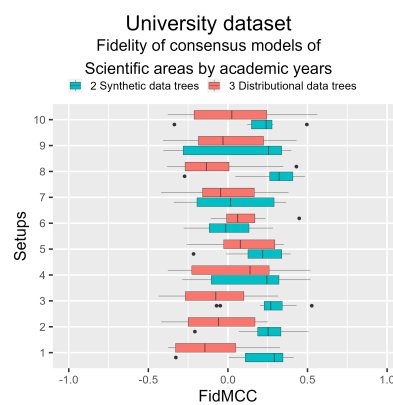
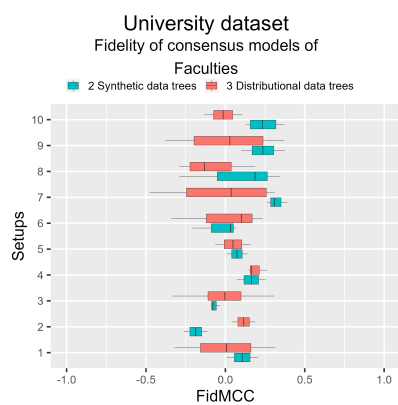
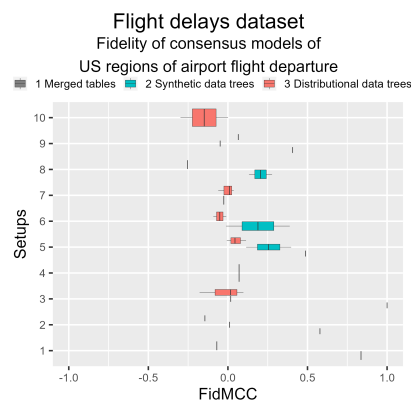
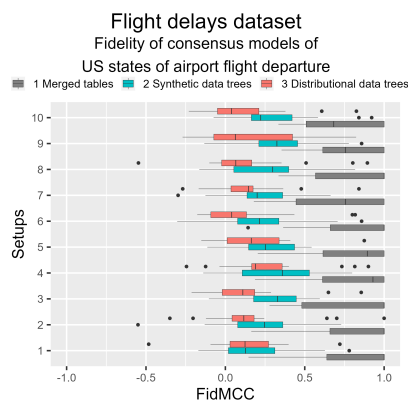
Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



C.4.4 Results of evaluation of Interpretability

The following tables present the average interpretability of consensus models across the five datasets, followed by boxplots of the metrics Size, Dimensionality, and Depth aggregated by strata and by setup.

Table C.24: Interpretability of consensus models from the Adult Census dataset

Consensus model	Representation	Size	Dimensionality	Depth
Asia	Merged table	9.200±5.115	6.900±2.914	
	Synthetic data tree	3.750±1.920	2.500±1.500	3.125±2.559
	Distributional data tree	3.000±1.225	2.000±1.225	2.000±1.225
Central America	Merged table	5.500±4.272	7.100±3.270	
	Synthetic data tree	3.667±3.727	1.833±1.863	2.500±3.354
	Distributional data tree	2.750±1.639	1.750±1.639	1.750±1.639
Europe	Merged table	8.200±3.945	8.500±2.693	
	Synthetic data tree	2.750±1.299	1.750±1.299	1.750±1.299
	Distributional data tree	2.500±0.866	1.500±0.866	1.500±0.866
North America	Merged table	7.800±2.960	8.400±2.245	
	Synthetic data tree	2.667±0.745	1.667±0.745	1.667±0.745
	Distributional data tree	2.600±0.800	1.600±0.800	1.600±0.800
South America	Merged table	9.900±5.974	7.200±2.135	
	Synthetic data tree	6.111±4.202	3.667±2.357	4.750±3.319
	Distributional data tree	4.750±1.714	3.375±1.495	3.250±1.199
Unknown	Merged table	10.800±5.269	7.500±1.565	
	Synthetic data tree	5.000±2.828	3.333±2.055	4.292±3.368
	Distributional data tree	3.714±1.979	2.571±1.841	2.429±1.678

Table C.25: Interpretability of consensus models from the Bank Marketing dataset

Consensus model	Representation	Size	Dimensionality	Depth
1st semester	Merged table	7.400±5.004	7.200±2.135	
	Synthetic data tree	9.200±11.548	2.600±1.625	4.000±3.795
	Distributional data tree	3.167±1.772	2.167±1.344	2.000±1.155
2nd semester	Merged table	8.800±6.493	6.000±2.757	
	Synthetic data tree	4.286±1.666	3.143±1.457	3.286±1.666
	Distributional data tree	2.714±0.700	1.714±0.700	1.714±0.700

Table C.26: Interpretability of consensus models from the Hotel Reservations dataset

Consensus model	Representation	Size	Dimensionality	Depth
2017-2nd semester	Merged table	8.300 ± 6.404	6.400 ± 4.200	
	Synthetic data tree	6.333 ± 2.625	4.667 ± 1.700	6.833 ± 2.779
	Distributional data tree	3.333 ± 0.943	2.333 ± 0.943	2.333 ± 0.943
2018-1st semester	Merged table	8.700 ± 6.357	10.000 ± 3.493	
	Synthetic data tree	6.000 ± 3.873	3.250 ± 2.046	5.188 ± 4.077
	Distributional data tree	2.875 ± 0.781	1.750 ± 0.661	1.750 ± 0.661
2018-2nd semester	Merged table	6.700 ± 5.569	6.600 ± 4.543	
	Synthetic data tree	4.400 ± 1.855	3.000 ± 1.414	3.400 ± 1.855
	Distributional data tree	2.600 ± 0.490	1.600 ± 0.490	1.600 ± 0.490

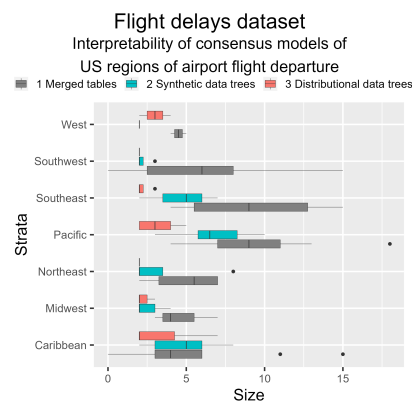
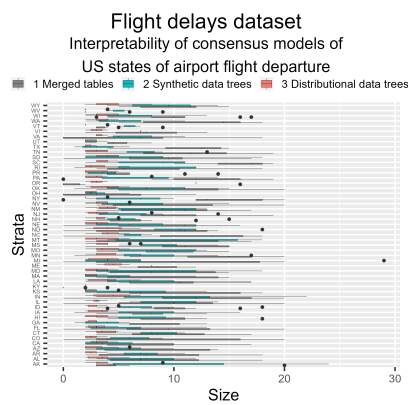
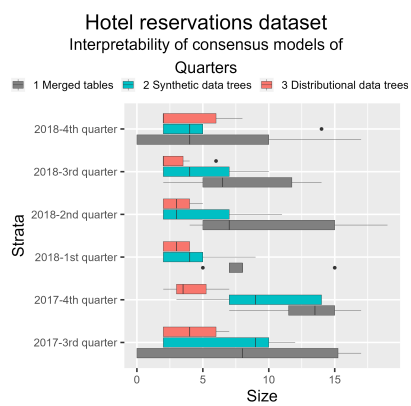
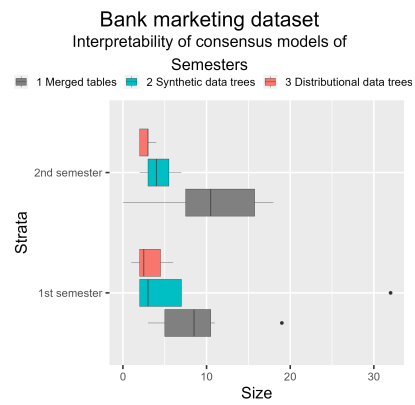
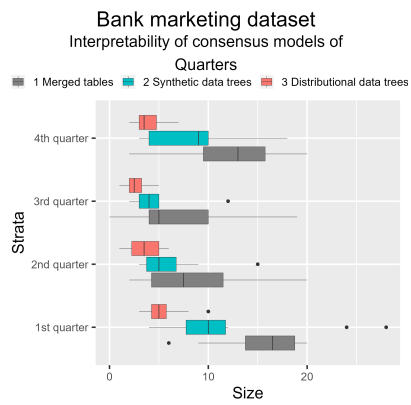
Table C.27: Interpretability of consensus models from the Flight Delays dataset

Consensus model	Representation	Size	Dimensionality	Depth
Caribbean	Merged table	5.000 ± 4.405	2.500 ± 1.360	
	Synthetic data tree	4.800 ± 2.135	2.600 ± 1.200	3.800 ± 2.135
	Distributional data tree	3.250 ± 1.785	1.875 ± 1.364	1.875 ± 1.166
Midwest	Merged table	2.600 ± 1.744	5.600 ± 0.490	
	Synthetic data tree	2.667 ± 0.943	1.667 ± 0.943	1.667 ± 0.943
	Distributional data tree	2.333 ± 0.471	1.333 ± 0.471	1.333 ± 0.471
Northeast	Merged table	5.700 ± 2.532	5.900 ± 0.300	
	Synthetic data tree	3.500 ± 2.598	1.250 ± 0.433	2.000 ± 1.732
	Distributional data tree	2.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Pacific	Merged table	9.000 ± 4.290	3.700 ± 0.900	
	Synthetic data tree	6.750 ± 2.107	2.750 ± 0.433	5.250 ± 2.165
	Distributional data tree	3.111 ± 0.994	2.000 ± 0.943	2.000 ± 0.943
Southeast	Merged table	5.700 ± 4.220	6.000 ± 0.000	
	Synthetic data tree	4.667 ± 2.055	2.000 ± 0.816	2.333 ± 1.247
	Distributional data tree	2.250 ± 0.433	1.250 ± 0.433	1.250 ± 0.433
Southwest	Merged table	5.300 ± 4.148	5.300 ± 1.792	
	Synthetic data tree	2.250 ± 0.433	1.250 ± 0.433	1.250 ± 0.433
	Distributional data tree	2.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
West	Merged table	2.500 ± 1.565	5.700 ± 0.458	
	Synthetic data tree	2.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	Distributional data tree	3.000 ± 1.000	2.000 ± 1.000	1.500 ± 0.500

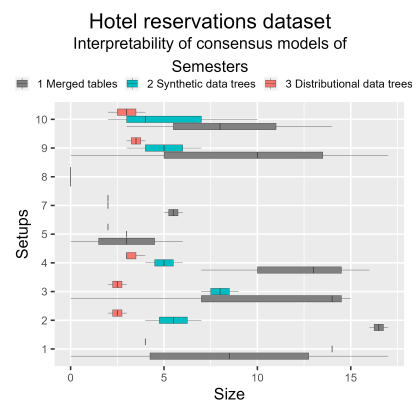
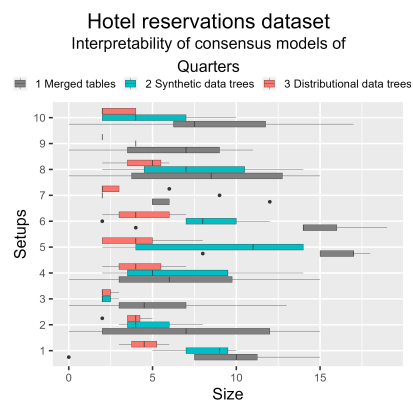
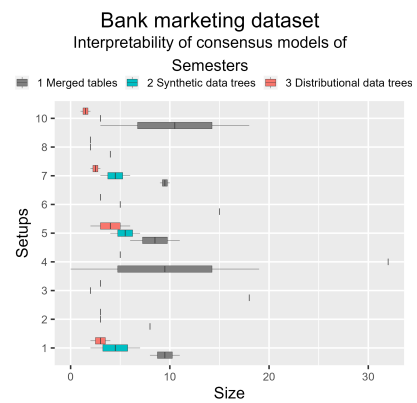
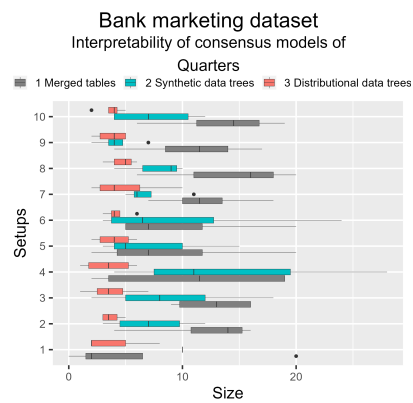
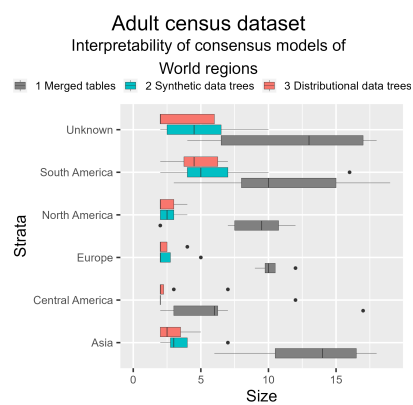
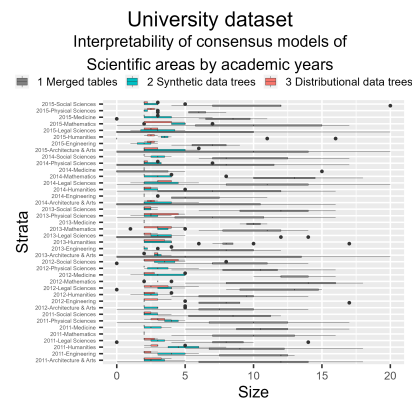
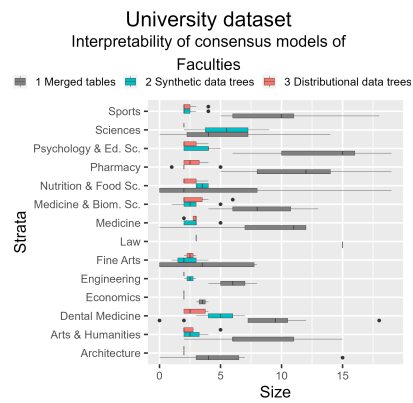
Table C.28: Interpretability of consensus models from the University of Porto dataset

Consensus model	Representation	Size	Dimensionality	Depth
Architecture	Merged table	5.900±5.718	18.600±6.359	
	Synthetic data tree	2.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	2.000±0.000	1.000±0.000	1.000±0.000
Arts & Humanities	Merged table	5.400±4.271	22.700±1.487	
	Synthetic data tree	2.750±0.829	1.500±0.500	1.750±0.829
	Distributional data tree	2.750±1.299	1.750±1.299	1.750±1.299
Dental Medicine	Merged table	8.800±5.212	14.800±5.115	
	Synthetic data tree	5.000±1.414	3.000±0.816	4.000±1.414
	Distributional data tree	2.833±0.898	1.667±0.745	1.833±0.898
Economics	Merged table	3.000±1.549	24.600±0.490	
	Synthetic data tree	2.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	2.000±0.000	1.000±0.000	1.000±0.000
Engineering	Merged table	2.600±2.010	24.400±0.663	
	Synthetic data tree	2.500±0.500	1.500±0.500	1.500±0.500
	Distributional data tree	2.000±0.000	1.000±0.000	1.000±0.000
Fine Arts	Merged table	7.600±6.422	15.000±9.849	
	Synthetic data tree	2.333±1.247	1.333±1.247	1.750±0.890
	Distributional data tree	2.500±0.500	1.000±0.000	1.500±0.500
Law	Merged table	6.700±4.670	23.300±0.640	
	Synthetic data tree	3.000±0.000	2.000±0.000	2.000±0.000
	Distributional data tree	n/a	n/a	n/a
Medicine	Merged table	9.100±3.833	18.600±6.280	
	Synthetic data tree	3.000±1.095	2.000±1.095	2.000±1.095
	Distributional data tree	2.750±0.433	1.750±0.433	1.750±0.433
Medic./Biom. Sc.	Merged table	6.300±3.796	22.600±1.020	
	Synthetic data tree	2.667±1.247	1.667±1.247	1.875±1.038
	Distributional data tree	3.000±1.528	1.333±0.471	1.833±1.213
Nutrition/Food Sc.	Merged table	5.900±6.472	11.000±9.066	
	Synthetic data tree	3.500±0.500	2.500±0.500	2.500±0.500
	Distributional data tree	2.600±0.800	1.600±0.800	1.600±0.800
Pharmacy	Merged table	8.900±4.969	24.100±0.700	
	Synthetic data tree	2.400±1.356	1.200±0.980	1.650±1.179
	Distributional data tree	2.750±0.829	1.500±0.866	1.750±0.829
Psychology/Ed. Sc.	Merged table	8.500±5.784	21.200±1.249	
	Synthetic data tree	3.400±1.200	2.000±0.894	2.400±1.200
	Distributional data tree	2.600±0.800	1.600±0.800	1.600±0.800
Sciences	Merged table	4.700±3.900	21.500±7.187	
	Synthetic data tree	5.500±3.500	2.500±1.500	3.000±2.000
	Distributional data tree	2.000±0.000	1.000±0.000	1.000±0.000
Sports	Merged table	8.200±4.707	23.300±1.552	
	Synthetic data tree	2.429±0.728	1.286±0.452	1.429±0.728
	Distributional data tree	2.429±0.728	1.429±0.728	1.429±0.728

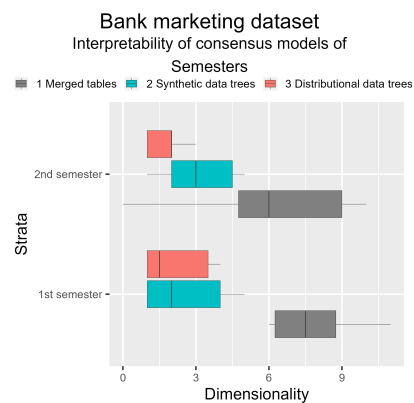
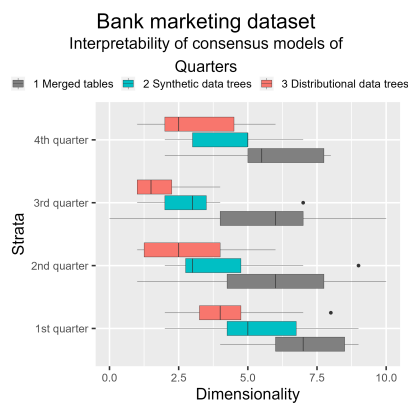
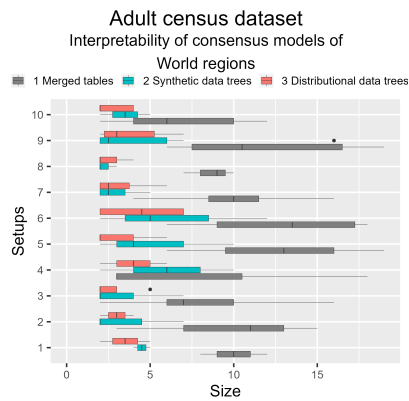
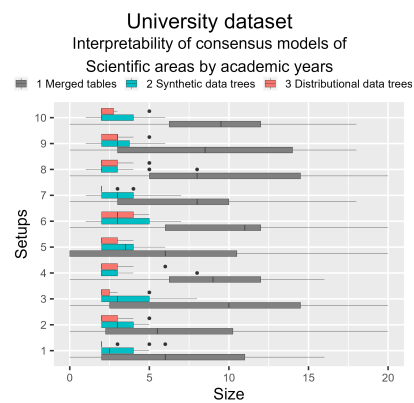
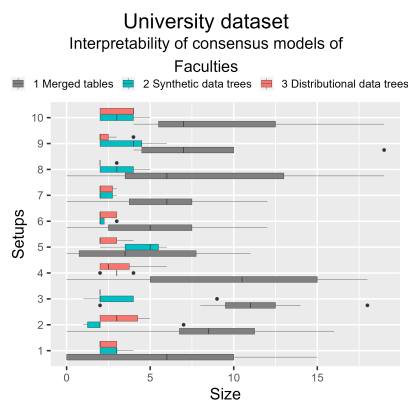
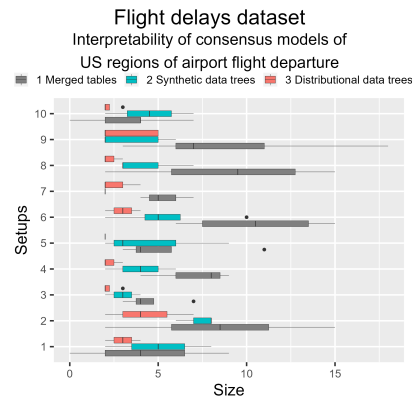
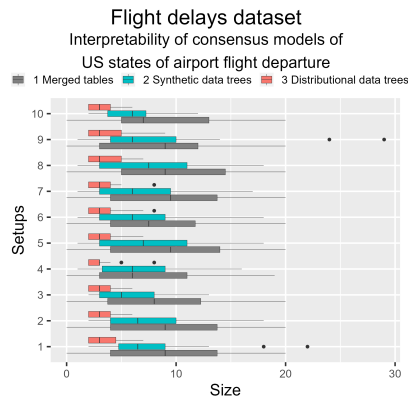
Datasets, Models and Evaluation Results



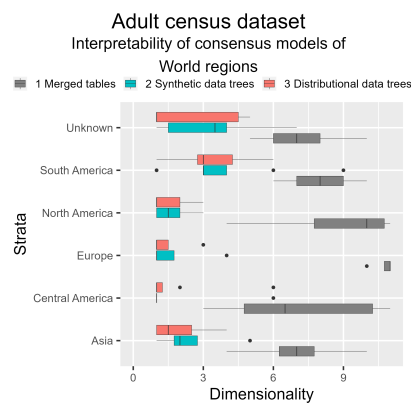
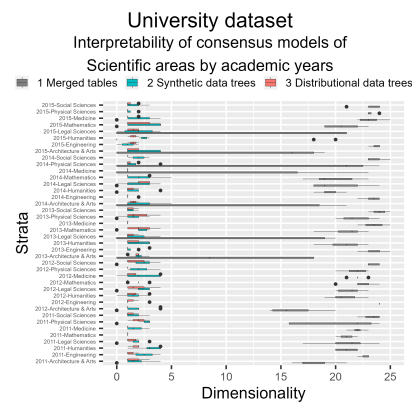
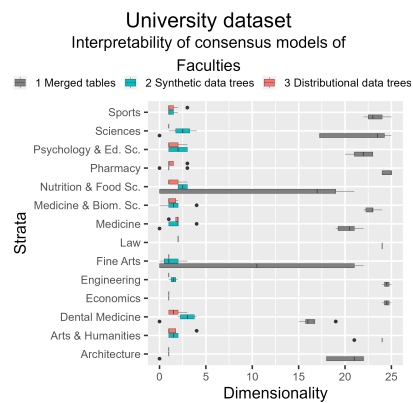
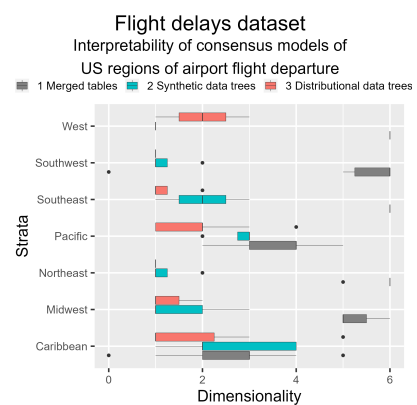
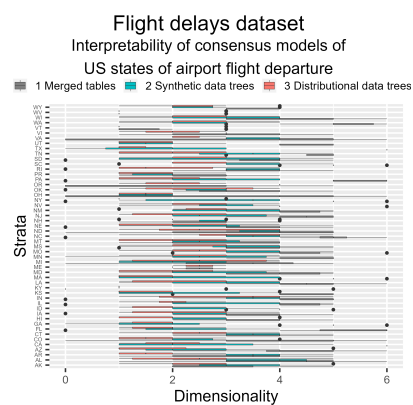
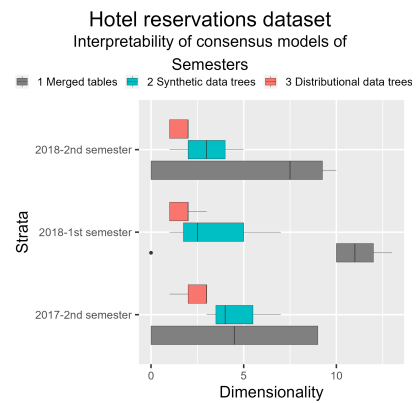
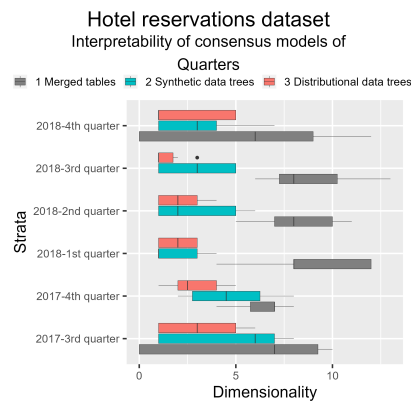
Datasets, Models and Evaluation Results



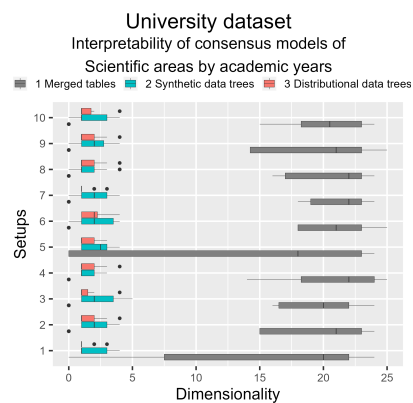
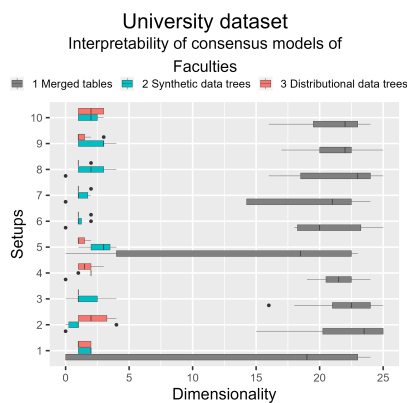
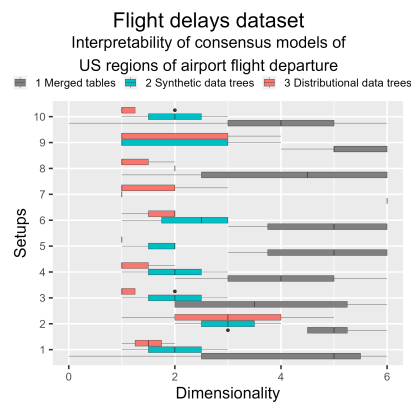
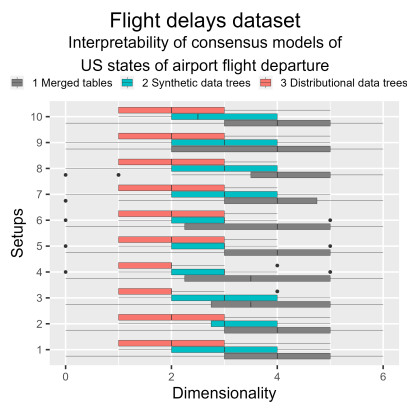
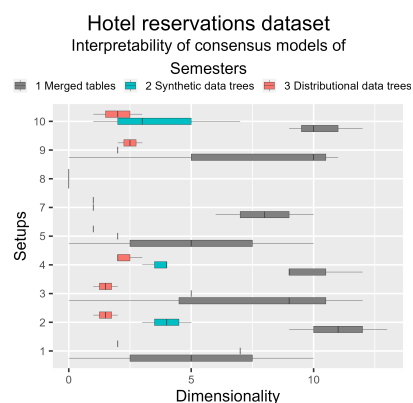
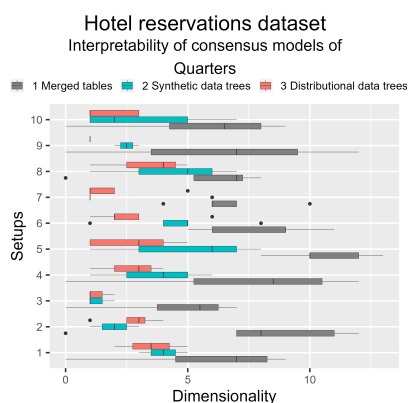
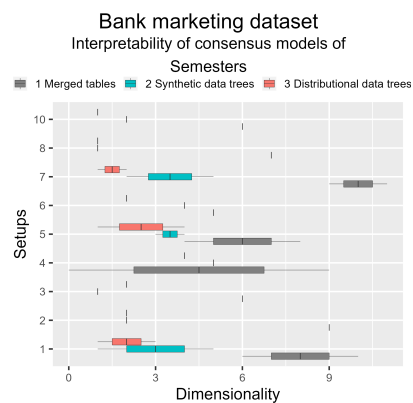
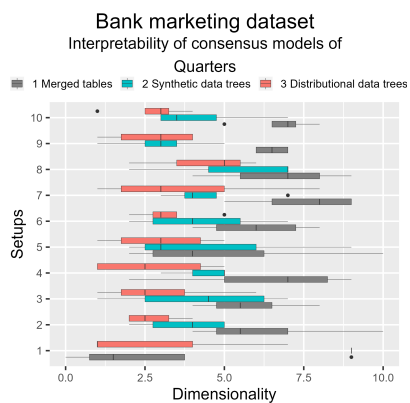
Datasets, Models and Evaluation Results



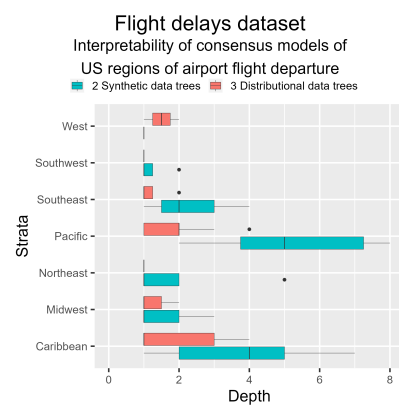
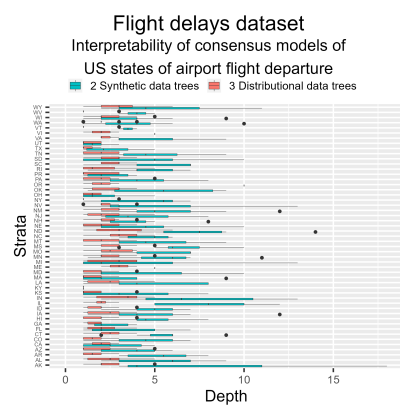
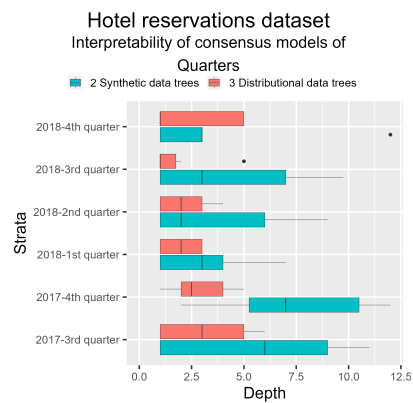
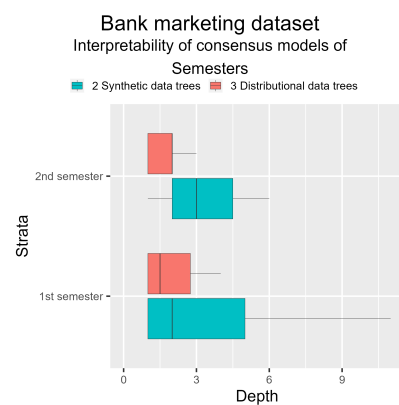
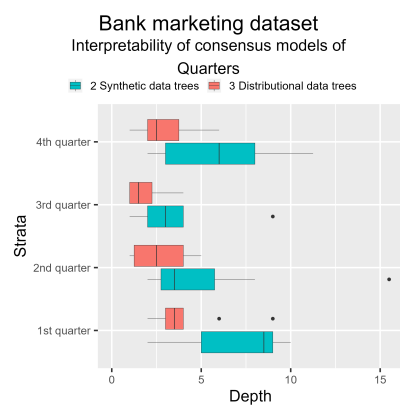
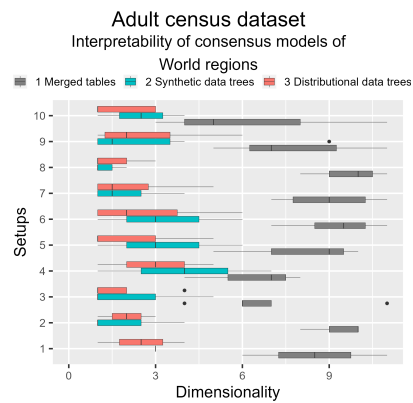
Datasets, Models and Evaluation Results



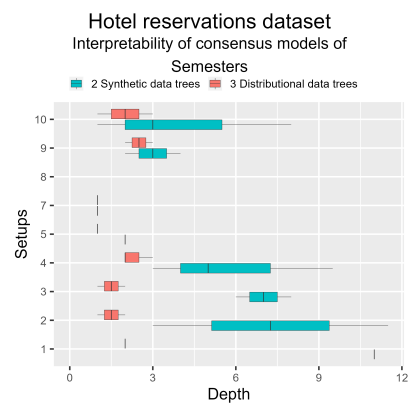
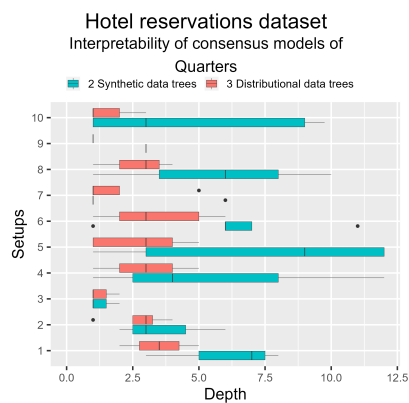
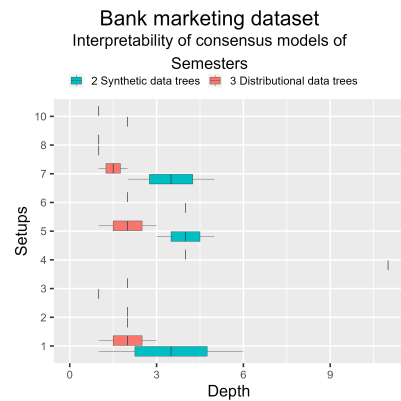
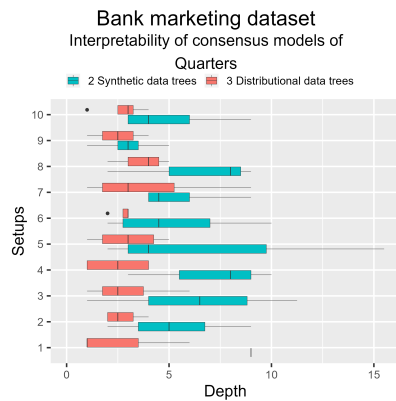
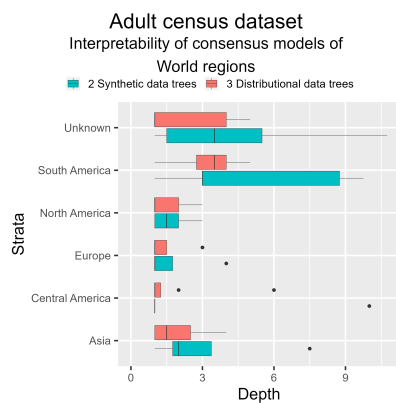
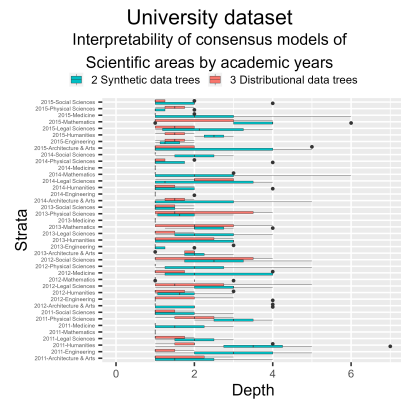
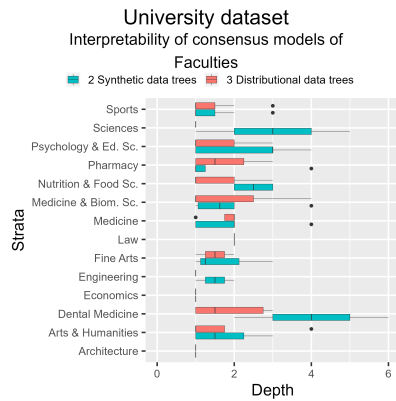
Datasets, Models and Evaluation Results



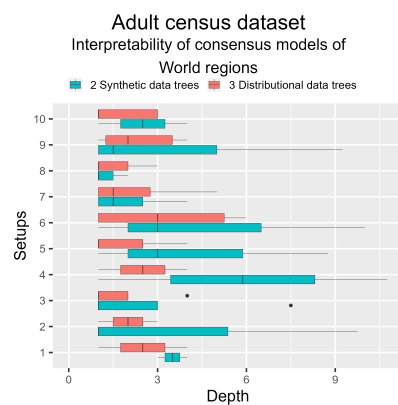
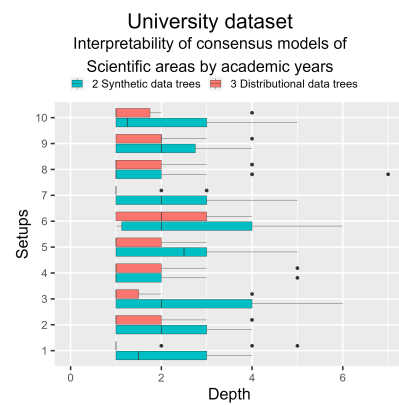
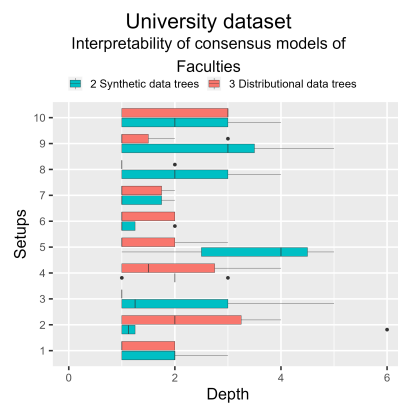
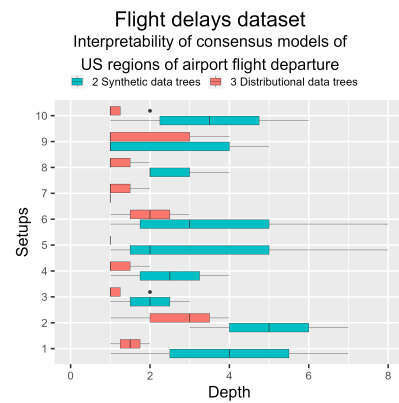
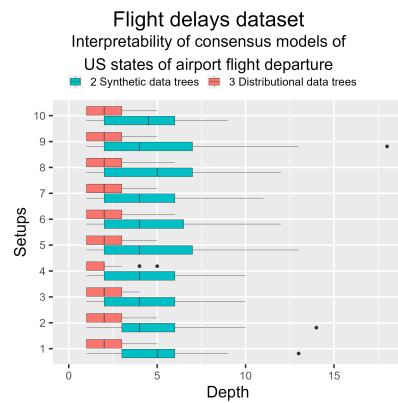
Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



C.4.5 Results of evaluation of Completeness

The following tables present the average completeness of consensus models across the five datasets, followed by boxplots of FracCovered (Fraction Covered), GComp (Geometrical Completeness), and DComp (Distributional Completeness) aggregated by strata and by setup.

Table C.29: Completeness of consensus models from the Adult Census dataset

Consensus model	Representation	FracCovered	GComp	DComp
Asia	Merged table	0.466±0.093	0.473±0.171	0.475±0.091
	Synthetic data tree	0.693±0.206	0.639±0.359	0.707±0.204
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Central America	Merged table	0.461±0.046	0.463±0.123	0.507±0.039
	Synthetic data tree	0.717±0.285	0.462±0.327	0.721±0.283
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Europe	Merged table	0.316±0.208	0.294±0.205	0.352±0.230
	Synthetic data tree	0.768±0.243	0.660±0.352	0.767±0.246
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
North America	Merged table	0.723±0.082	0.804±0.074	0.683±0.099
	Synthetic data tree	0.850±0.089	0.649±0.348	0.847±0.087
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
South America	Merged table	0.781±0.148	0.823±0.156	0.744±0.165
	Synthetic data tree	0.719±0.267	0.648±0.345	0.743±0.250
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Unknown	Merged table	0.796±0.117	0.708±0.060	0.836±0.091
	Synthetic data tree	0.711±0.142	0.446±0.285	0.729±0.127
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000

Table C.30: Completeness of consensus models from the Bank Marketing dataset

Consensus model	Representation	FracCovered	GComp	DComp
1st semester	Merged table	0.628±0.251	0.268±0.211	0.664±0.256
	Synthetic data tree	0.906±0.078	0.737±0.094	0.909±0.093
	Distributional data tree	0.981±0.043	1.000±0.000	0.981±0.043
2nd semester	Merged table	0.326±0.238	0.201±0.277	0.386±0.230
	Synthetic data tree	0.880±0.192	0.894±0.160	0.916±0.152
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000

Table C.31: Completeness of consensus models from the Hotel Reservations dataset

Consensus model	Representation	FracCovered	GComp	DComp
2017-2nd semester	Merged table	0.247±0.147	0.011±0.014	0.199±0.125
	Synthetic data tree	0.968±0.031	0.794±0.170	0.953±0.051
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
2018-1st semester	Merged table	0.050±0.057	0.000±0.000	0.036±0.048
	Synthetic data tree	0.978±0.034	0.905±0.108	0.983±0.024
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
2018-2nd semester	Merged table	0.189±0.251	0.059±0.161	0.193±0.258
	Synthetic data tree	0.987±0.021	0.925±0.101	0.988±0.020
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000

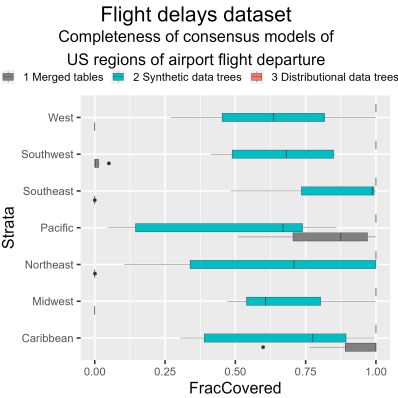
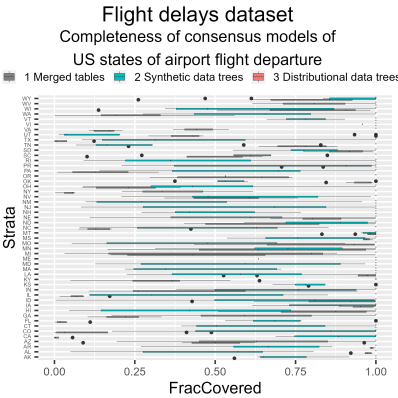
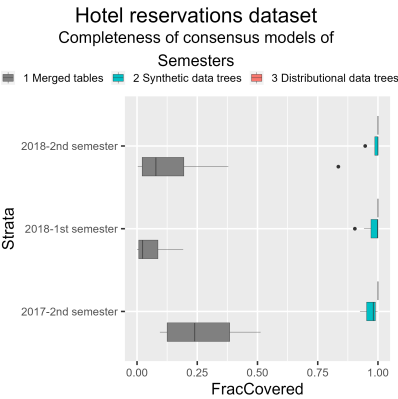
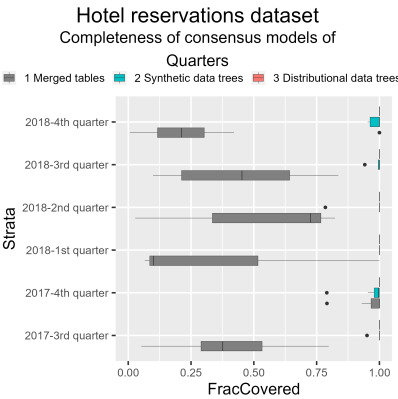
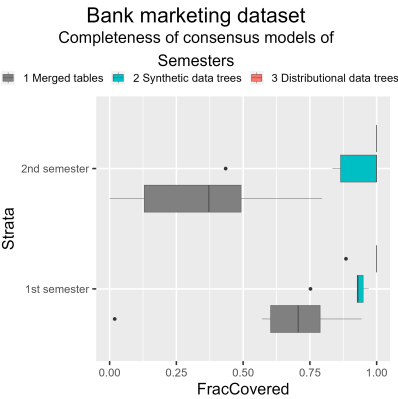
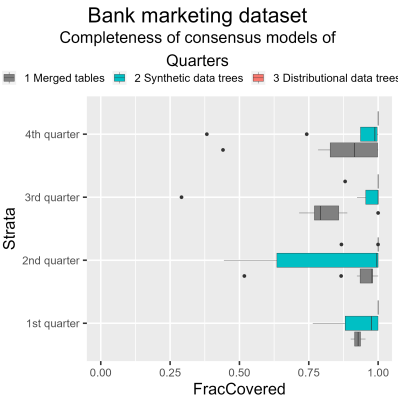
Table C.32: Completeness of consensus models from the Flight Delays dataset

Consensus model	Representation	FracCovered	GComp	DComp
Caribbean	Merged table	0.921±0.129	0.358±0.327	0.448±0.251
	Synthetic data tree	0.671±0.275	0.560±0.345	0.642±0.226
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Midwest	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	0.693±0.224	0.675±0.234	0.632±0.285
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Northeast	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	0.630±0.386	0.516±0.484	0.577±0.424
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Pacific	Merged table	0.775±0.203	0.367±0.136	0.396±0.100
	Synthetic data tree	0.507±0.321	0.308±0.235	0.343±0.212
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Southeast	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	0.823±0.241	0.805±0.265	0.817±0.243
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Southwest	Merged table	0.014±0.021	0.001±0.003	0.004±0.006
	Synthetic data tree	0.657±0.197	0.536±0.351	0.561±0.301
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
West	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	0.635±0.365	0.643±0.357	0.644±0.356
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000

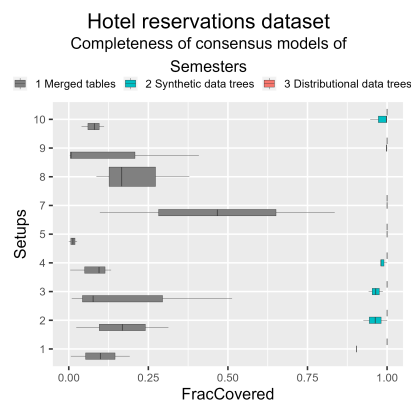
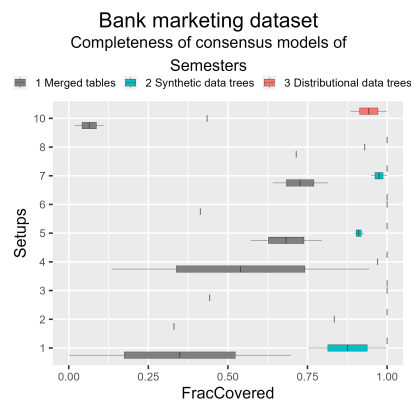
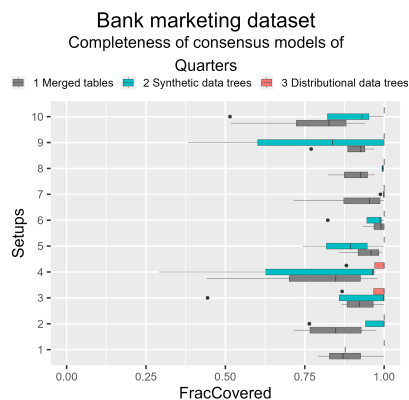
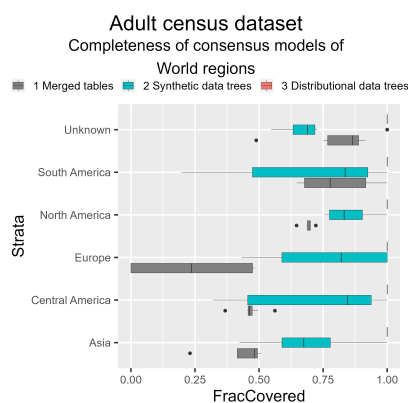
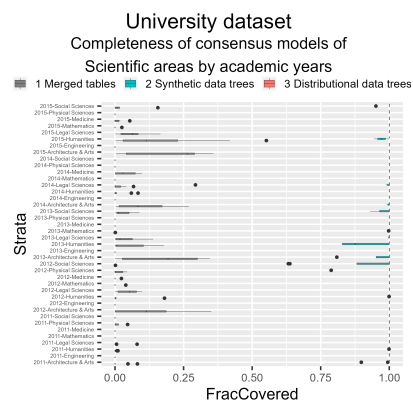
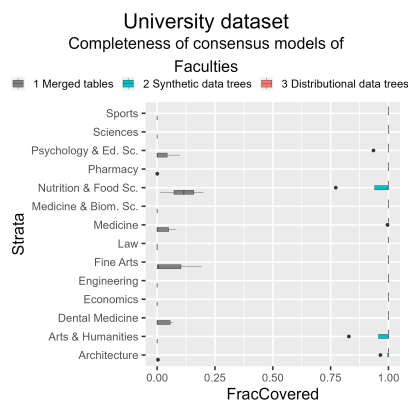
Table C.33: Completeness of consensus models from the University of Porto dataset

Consensus model	Representation	FracCovered	GComp	DComp
Architecture	Merged table	0.004±0.010	0.000±0.000	0.000±0.000
	Synthetic data tree	0.992±0.014	0.724±0.339	0.979±0.040
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Arts & Humanities	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	0.957±0.074	0.917±0.144	0.940±0.104
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Dental Medicine	Merged table	0.020±0.028	0.000±0.000	0.005±0.014
	Synthetic data tree	1.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Economics	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Engineering	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	0.750±0.250	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Fine Arts	Merged table	0.055±0.070	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	0.667±0.471
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Law	Merged table	0.001±0.003	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	n/a	n/a	n/a
Medicine	Merged table	0.039±0.041	0.000±0.000	0.000±0.000
	Synthetic data tree	0.999±0.002	0.984±0.032	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Medic./Biom. Sc.	Merged table	0.003±0.008	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	0.833±0.373
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Nutrition/Food Sc.	Merged table	0.115±0.062	0.000±0.000	0.009±0.024
	Synthetic data tree	0.942±0.098	0.690±0.325	0.994±0.009
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Pharmacy	Merged table	0.006±0.018	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	0.800±0.400
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Psychology/Ed. Sc.	Merged table	0.037±0.042	0.000±0.000	0.001±0.004
	Synthetic data tree	0.987±0.026	0.933±0.133	0.994±0.012
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Sciences	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000
Sports	Merged table	0.000±0.000	0.000±0.000	0.000±0.000
	Synthetic data tree	1.000±0.000	1.000±0.000	1.000±0.000
	Distributional data tree	1.000±0.000	1.000±0.000	1.000±0.000

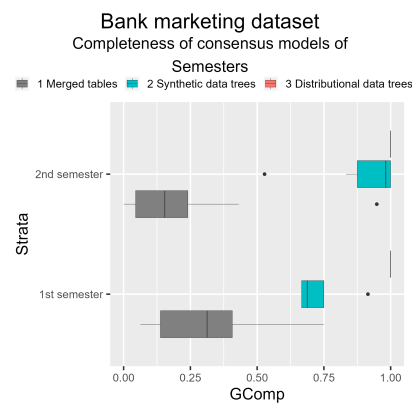
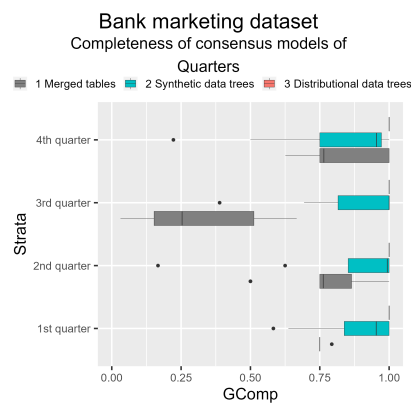
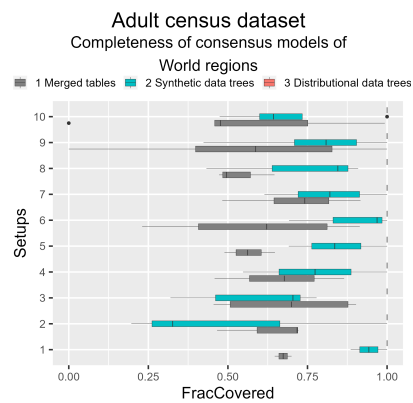
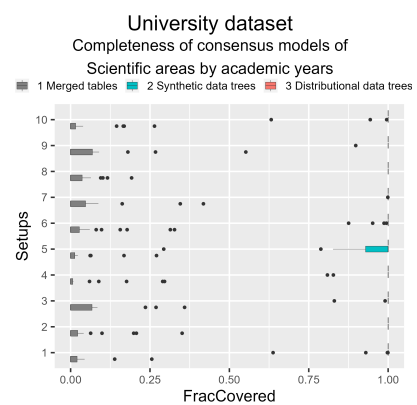
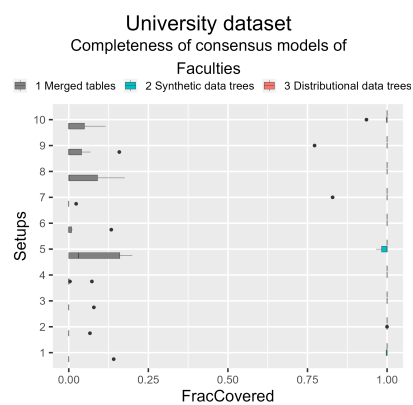
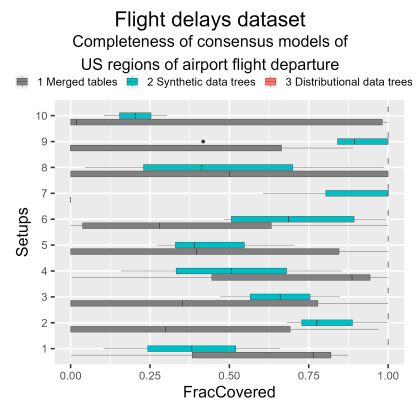
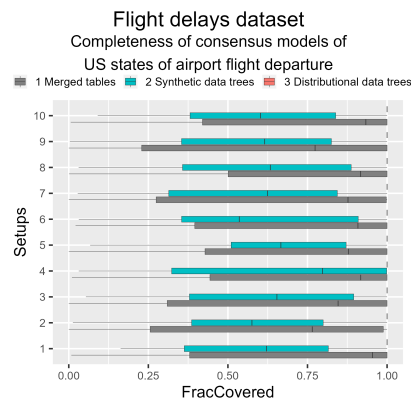
Datasets, Models and Evaluation Results



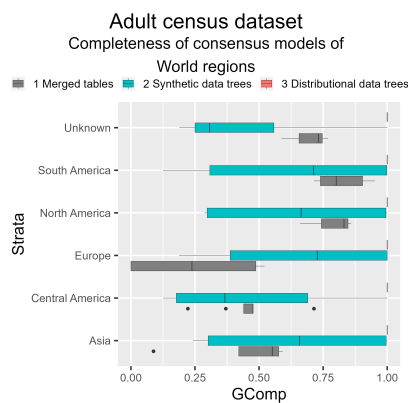
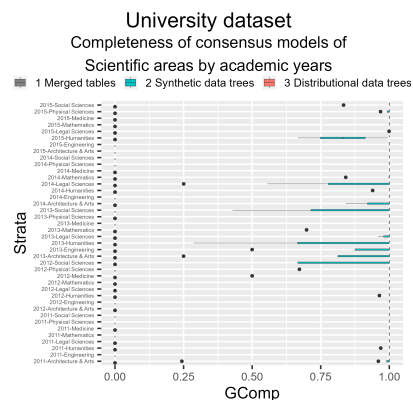
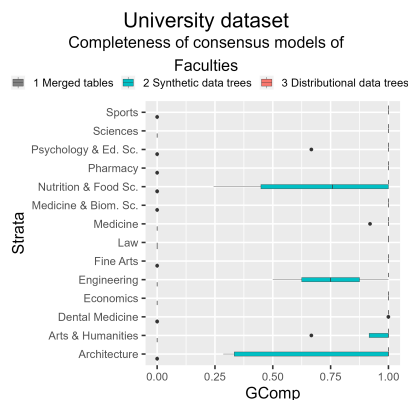
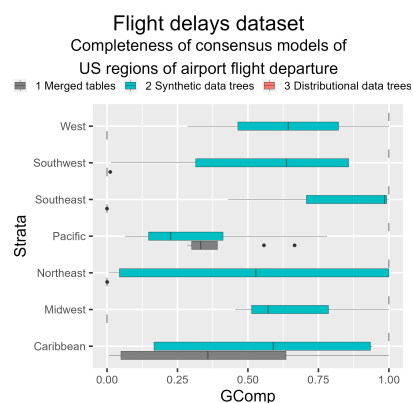
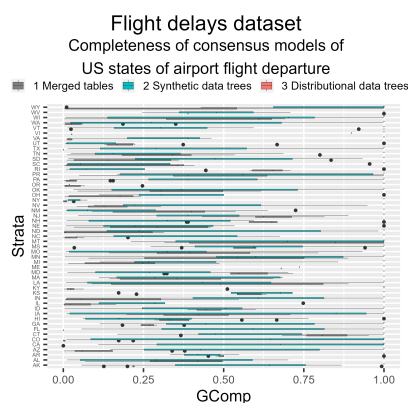
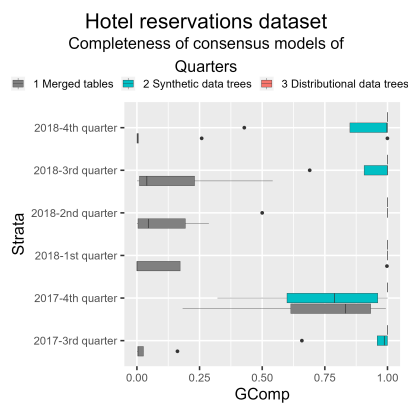
Datasets, Models and Evaluation Results



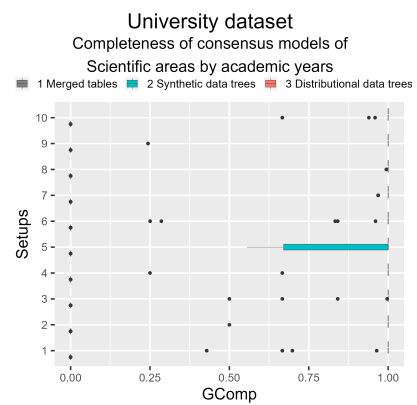
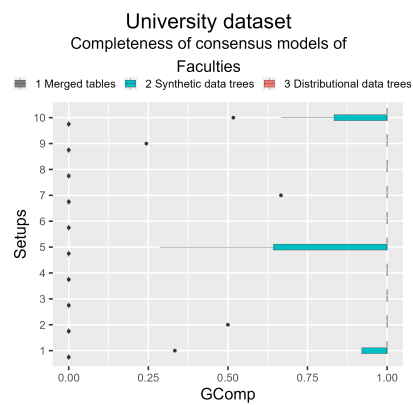
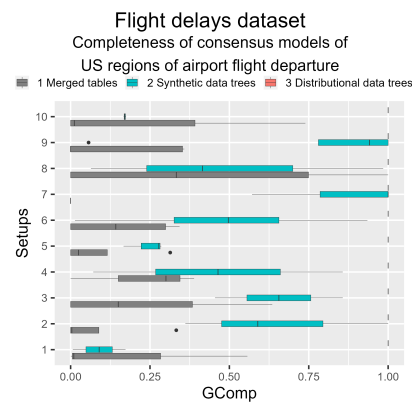
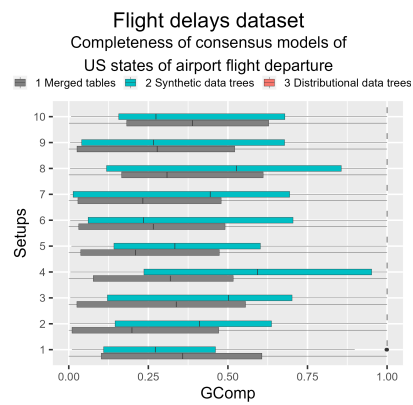
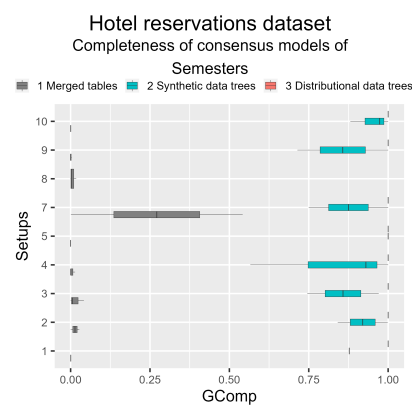
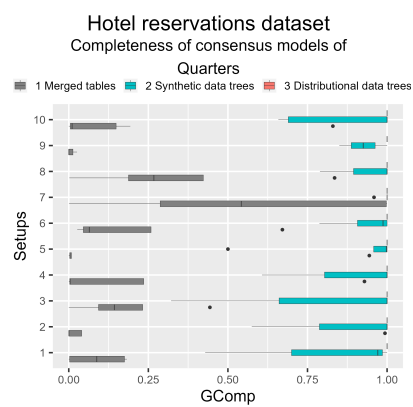
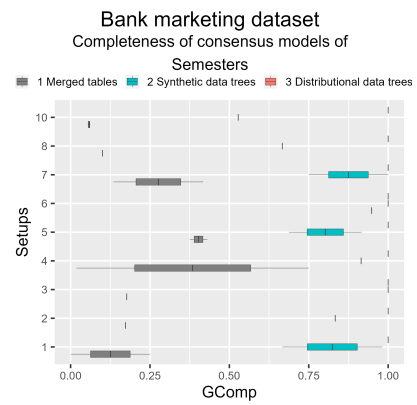
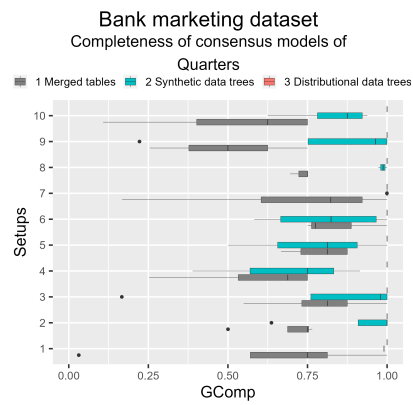
Datasets, Models and Evaluation Results



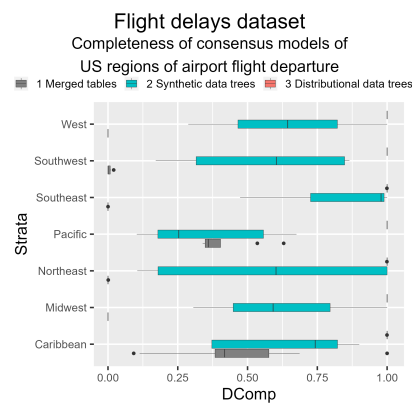
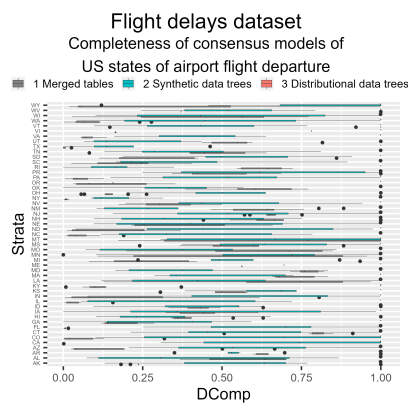
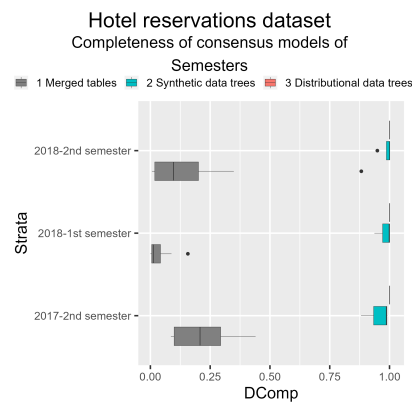
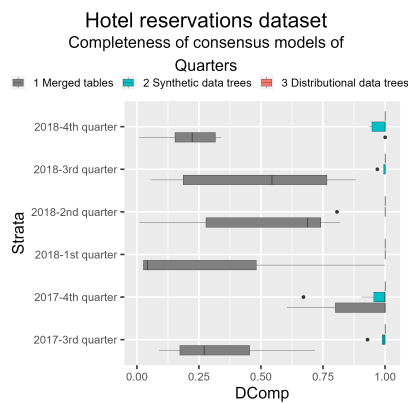
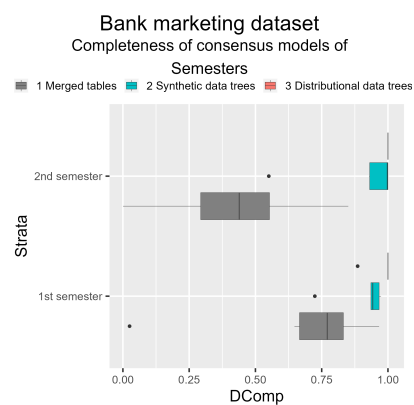
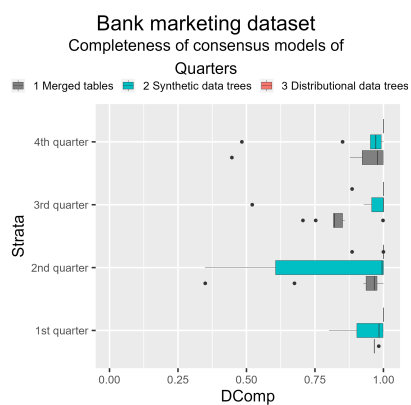
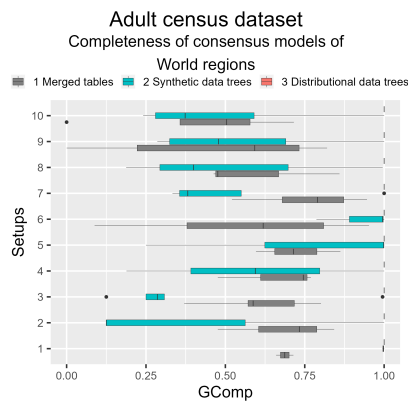
Datasets, Models and Evaluation Results



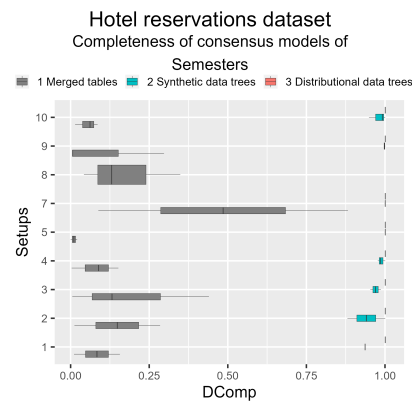
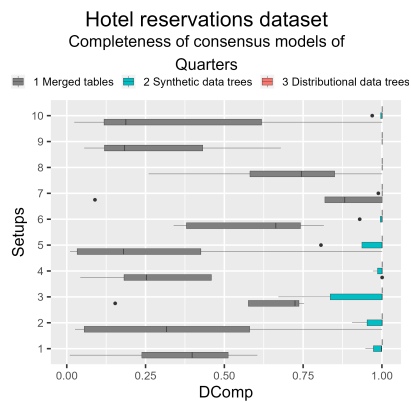
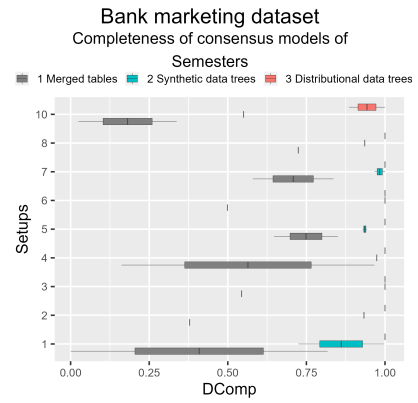
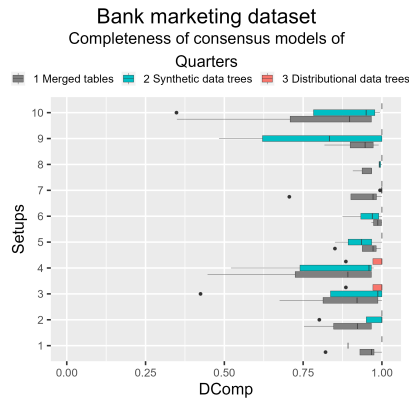
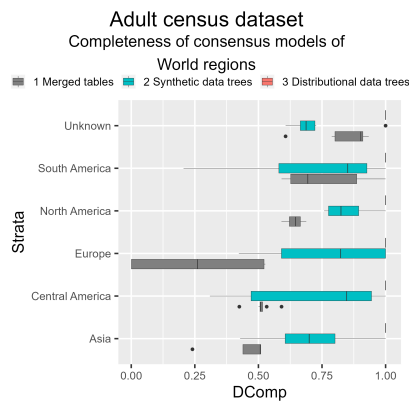
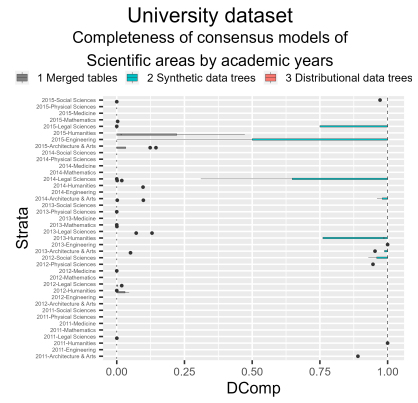
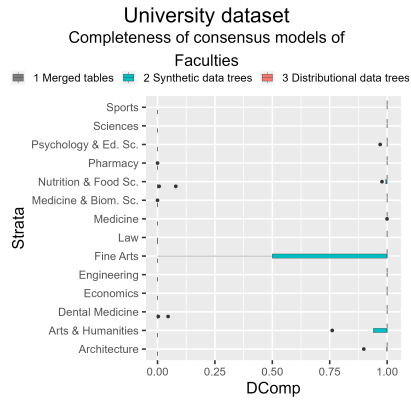
Datasets, Models and Evaluation Results



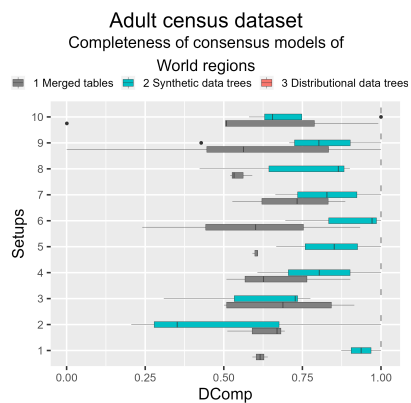
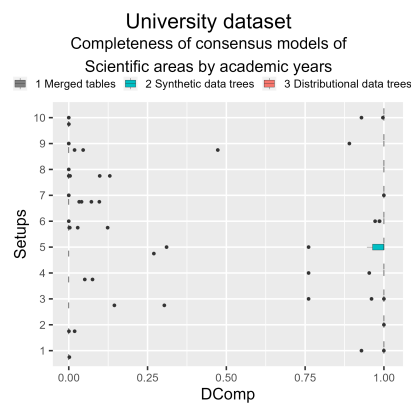
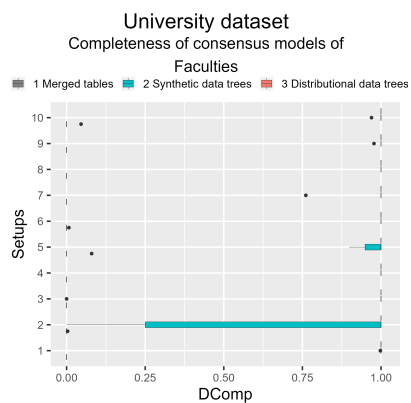
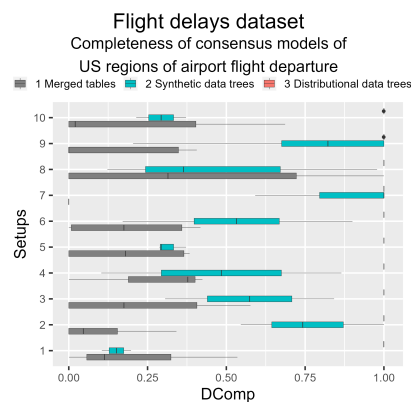
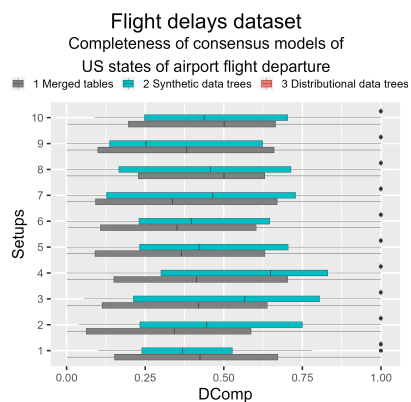
Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



Datasets, Models and Evaluation Results



References

- Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 9 2018. ISSN 21693536. doi: [10.1109/ACCESS.2018.2870052](https://doi.org/10.1109/ACCESS.2018.2870052).
- R. J. Adler, D. Firmin, and D. G. Kendall. A non-gaussian model for random surfaces. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 303(1479):433–462, 12 1981. ISSN 0080-4614.
- Ronay Ak, Valeria Vitelli, and Enrico Zio. An Interval-Valued Neural Network Approach for Uncertainty Quantification in Short-Term Wind Speed Prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11):2787–2800, 11 2015. ISSN 21622388. doi: [10.1109/TNNLS.2015.2396933](https://doi.org/10.1109/TNNLS.2015.2396933).
- Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, 6 1978. ISSN 0018-9340. doi: [10.1109/TC.1978.1675141](https://doi.org/10.1109/TC.1978.1675141).
- Noor Sami Al-Anbaki, Nadim Obeid, and Khair Eddin. A Defeasible Logic-based Framework for Contextualizing Deployed Applications. *International Journal of Advanced Computer Science and Applications*, 10(9), 2019. ISSN 21565570. doi: [10.14569/IJACSA.2019.0100923](https://doi.org/10.14569/IJACSA.2019.0100923).
- Baligh Al-Helali, Qi Chen, Bing Xue, and Mengjie Zhang. A new imputation method based on genetic programming and weighted KNN for symbolic regression with incomplete data. *Soft Computing*, 25(8):5993–6012, 4 2021. ISSN 1432-7643. doi: [10.1007/s00500-021-05590-y](https://doi.org/10.1007/s00500-021-05590-y).
- Yousra Abdul Alsahib S. Aldeen, Mazleena Salleh, and Mohammad Abdur Razzaque. A comprehensive review on privacy preserving data mining. *SpringerPlus*, 4(1):1–36, 12 2015. ISSN 21931801. doi: [10.1186/s40064-015-1481-x](https://doi.org/10.1186/s40064-015-1481-x).
- Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, José María Alonso Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information Fusion*, 99:101805, 2023. ISSN 1566-2535. doi: <https://doi.org/10.1016/j.inffus.2023.101805>.
- Hussein Almuallim. An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence*, 83(2):347–362, 6 1996. ISSN 00043702. doi: [10.1016/0004-3702\(95\)00060-7](https://doi.org/10.1016/0004-3702(95)00060-7).
- Khaled Alomar, Halil Ibrahim Aysel, and Xiaohao Cai. Data Augmentation in Classification and Segmentation: A Survey and New Strategies. *Journal of Imaging*, 9(2), 2 2023. ISSN 2313433X. doi: [10.3390/jimaging9020046](https://doi.org/10.3390/jimaging9020046).
- José María Alonso Moral. *Interpretable fuzzy systems modeling with cooperation between expert and induced knowledge*. PhD thesis, Telecomunicacion, October 2007.

References

- José María Alonso Moral and Luis Magdalena. HILK++: An interpretability-guided fuzzy modeling methodology for learning readable and comprehensible fuzzy rule-based classifiers. *Soft Computing*, 15(10):1959–1980, 2011. doi: [10.1007/s00500-010-0628-5](https://doi.org/10.1007/s00500-010-0628-5).
- José María Alonso Moral, Luis Magdalena, and Serge Guillaume. HILK: A new methodology for designing highly interpretable linguistic knowledge bases using the fuzzy logic formalism. *International Journal of Intelligent Systems*, 23(7):761–794, 2008. doi: <https://doi.org/10.1002/int.20288>.
- Artur Andrzejak, Felix Langner, and Silvestre Zabala. Interpretable Models from Distributed Data via Merging of Decision Trees. *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2013.
- Cecilio Angulo, Davide Anguita, and Luis González. Interval discriminant analysis using Support Vector Machines. In *European Symposium on Artificial Neural Networks*, page 593. D-side, 2007. ISBN 2930307072.
- Pedro Antunes, Valeria Herskovic, Sergio F. Ochoa, and Jose A. Pino. Structuring dimensions for collaborative systems evaluation. *ACM Computing Surveys*, 44(2):1–28, 2 2012. ISSN 0360-0300. doi: [10.1145/2089125.2089128](https://doi.org/10.1145/2089125.2089128).
- Annalisa Appice, Claudia D’Amato, Floriana Esposito, and Donato Malerba. Classification of symbolic objects: A lazy learning approach. *Intelligent Data Analysis*, 10(4):301–324, 2006. ISSN 15714128. doi: [10.3233/ida-2006-10402](https://doi.org/10.3233/ida-2006-10402).
- Ibrahim Berkan Aydilek and Ahmet Arslan. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, 233:25–35, 6 2013. ISSN 00200255. doi: [10.1016/j.ins.2013.01.021](https://doi.org/10.1016/j.ins.2013.01.021).
- Olawale F. Ayilara, Lisa Zhang, Tolulope T. Sajobi, Richard Sawatzky, Eric Bohm, and Lisa M. Lix. Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry. *Health and Quality of Life Outcomes*, 17(1):106, 12 2019. ISSN 1477-7525. doi: [10.1186/s12955-019-1181-2](https://doi.org/10.1186/s12955-019-1181-2).
- Nahla Hosny Barakat and Joachim Diederich. Eclectic Rule-Extraction from Support Vector Machines. *Int. J. Comput. Intell.*, 2(1):59–62, 2005.
- Amanda N. Baraldi and Craig K. Enders. An introduction to modern missing data analyses. *Journal of School Psychology*, 48(1):5–37, 2 2010. ISSN 00224405. doi: [10.1016/j.jsp.2009.10.001](https://doi.org/10.1016/j.jsp.2009.10.001).
- Elias Bareinboim and Jin Tian. Recovering Causal Effects from Selection Bias. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3475–3481, 2015.
- Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 6 2020. ISSN 15662535. doi: [10.1016/j.inffus.2019.12.012](https://doi.org/10.1016/j.inffus.2019.12.012).
- Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via Model Extraction, 6 2017.
- Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. Sirius: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, 15(1):427–505, 2021. ISSN 19357524. doi: [10.1214/20-EJS1792](https://doi.org/10.1214/20-EJS1792).

References

- Adrien Bibal and Benoît Frénay. Interpretability of Machine Learning Models and Representations: an Introduction. In *24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2016)*, 2016. ISBN 9782875870278.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006.
- Wayne D. Blizard. The Development of Multiset Theory. *Modern Logic*, 4(1):319–352, 1991.
- Hans-Hermann Bock and Edwin Diday. *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*. Springer, Berlin/Heidelberg, 2000.
- A. Bowman and A. Azzalini. R package sm: nonparametric smoothing methods, 2010.
- Joshua Brady and Donald R Brockmeier. Bias-Variance Tradeoff: A Property-Casualty Modeler’s Perspective. *Variance: Advancing the Science of Risk*, 13(2):207–232, 2018.
- Max Bramer. *Principles of Data Mining*. Springer London, London, 2007. ISBN 978-1-84628-765-7. doi: [10.1007/978-1-84628-766-4](https://doi.org/10.1007/978-1-84628-766-4).
- Leo Breiman. Bagging Predictors. *Machine Learning*, 24:123–140, 1996.
- Leo Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.
- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification And Regression Trees*. Chapman and Hall/CRC, 1984. ISBN 9781315139470. doi: [10.1201/9781315139470](https://doi.org/10.1201/9781315139470).
- Paula Brito. Symbolic Data Analysis: Another look at the interaction of Data Mining and Statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):281–295, 2014. ISSN 19424795. doi: [10.1002/widm.1133](https://doi.org/10.1002/widm.1133).
- Randal E. Bryant. Binary Decision Diagrams. In *Handbook of Model Checking*, pages 191–217. Springer International Publishing, Cham, 2018. doi: [10.1007/978-3-319-10575-8_7](https://doi.org/10.1007/978-3-319-10575-8_7).
- N. Burkart and M. F. Huber. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, 70(January):245–317, 2021. ISSN 10769757. doi: [10.1613/JAIR.1.12228](https://doi.org/10.1613/JAIR.1.12228).
- B. Bursteinas and J. Long. Merging Distributed Classifiers. *5th World Multi- conference on Systemics, Cybernetics and Informatics*, 2001.
- R. G. Carpenter and J. L. Emery. Identification and follow-up of infants at risk of sudden death in infancy. *Nature*, 250:729–729, 1974. doi: [10.1038/250729a0](https://doi.org/10.1038/250729a0).
- Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics (section: Artificial Intelligence)*, pages 1–34, 2019. ISSN 20799292. doi: [10.3390/electronics8080832](https://doi.org/10.3390/electronics8080832).
- P. Cazes, A. Chouakria, E. Diday, and Y. Schektman. Extension de l’analyse en composantes principales à des données de type intervalle. *Rev. Statistique Appliquée*, 1, 1997.
- Jadzia Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 10 1987. ISSN 00207373. doi: [10.1016/S0020-7373\(87\)80003-2](https://doi.org/10.1016/S0020-7373(87)80003-2).

References

- Philip K. Chan and Salvatore J. Stolfo. On the Accuracy of Meta-learning for Scalable Data Mining. *Journal of Intelligent Information Systems*, 8:5–28, 1997.
- Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, New York, NY, USA, 8 2016. ACM. ISBN 9781450342322. doi: [10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785).
- Yen Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics and Epidemiology*, 1(1):161–187, 1 2017. ISSN 24709379.
- Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1), 1 2020. ISSN 14712164. doi: [10.1186/s12864-019-6413-7](https://doi.org/10.1186/s12864-019-6413-7).
- Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen’s Kappa and Brier Score in Binary Classification Assessment. *IEEE Access*, 9:78368–78381, 2021. ISSN 21693536. doi: [10.1109/ACCESS.2021.3084050](https://doi.org/10.1109/ACCESS.2021.3084050).
- Peter Clark and Robin Boswell. Rule Induction with CN2: Some Recent Improvements. In *Machine Learning – Proc. Fifth European Conf. (EWSL-91)*, pages 151–163, 1991.
- William W Cohen. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.
- David Collier. Translating Quantitative Methods for Qualitative Researchers: The case of selection bias. *American Political Science Review*, 89(2), 1995.
- Roberto Confalonieri, Tillman Weyde, Tarek R Besold, Fermín Moscoso, and Del Prado Martín. TREPAN Reloaded: A Knowledge-driven Approach to Explaining Black-box Models. In *24th European Conference on Artificial Intelligence (ECAI 2020)*, 2020.
- Giorgio Corani, Alessandro Antonucci, Denis D. Mauá, and Sandra Gabaglio. Trading off Speed and Accuracy in Multilabel Classification. In *European Workshop on Probabilistic Graphical Models*, pages 145–159. Springer, Cham, 2014. doi: [10.1007/978-3-319-11433-0_10](https://doi.org/10.1007/978-3-319-11433-0_10).
- Mark W. Craven and Jude W. Shavlik. Extracting Tree-Structured Representations of Trained Networks. *Advances in Neural Information Processing Systems*, 8, 1996.
- Nicole Cruz, Jean Baratgin, Mike Oaksford, and David E. Over. Bayesian reasoning with ifs and ands and ors. *Frontiers in Psychology*, 6, 2 2015. ISSN 1664-1078. doi: [10.3389/fpsyg.2015.00192](https://doi.org/10.3389/fpsyg.2015.00192).
- Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. ENDER: a statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery*, 21(1):52–90, 7 2010. ISSN 1384-5810. doi: [10.1007/s10618-010-0177-7](https://doi.org/10.1007/s10618-010-0177-7).
- Sónia Dias and Paula Brito. Distribution and Symmetric Distribution Regression Model for Interval-Valued Variables. *arXiv*, 4 2013.
- Edwin Diday. Principal Component Analysis for Categorical Histogram Data: Some Open Directions of Research. In *Classification and Multivariate Analysis for Complex Data Structures*, pages 3–15. Springer, Berlin, 2011. doi: [10.1007/978-3-642-13312-1_1](https://doi.org/10.1007/978-3-642-13312-1_1).

References

- Edwin Diday. Thinking by classes in data science: the symbolic data analysis paradigm. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(5):172–205, 9 2016. ISSN 19390068. doi: [10.1002/wics.1384](https://doi.org/10.1002/wics.1384).
- Pedro Domingos. The Role of Occam’s Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.
- A. Rogier T. Donders, Geert J.M.G. van der Heijden, Theo Stijnen, and Karel G.M. Moons. Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59(10):1087–1091, 10 2006. ISSN 08954356. doi: [10.1016/j.jclinepi.2006.01.014](https://doi.org/10.1016/j.jclinepi.2006.01.014).
- Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning. *Springer Series on Challenges in Machine Learning*, 1(Explainable and Interpretable Models in Computer Vision and Machine Learning):1–13, 2017.
- Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 1 2020. ISSN 15577317. doi: [10.1145/3359786](https://doi.org/10.1145/3359786).
- Heetika Duggal and Parminder Singh. Comparative Study of the Performance of M5-Rules Algorithm with Different Algorithms. *Journal of Software Engineering and Applications*, 05(04): 270–276, 2012. ISSN 1945-3116. doi: [10.4236/jsea.2012.54032](https://doi.org/10.4236/jsea.2012.54032).
- T. Duong. ks: Kernel smoothing, 2011.
- Prarthana Dutta, Naresh Babu Muppalaneni, and Ripon Patgiri. A Survey on Explainability: Why Should We Believe the Accuracy of A Model? *Journal of Artificial Intelligence Research*, 1 (April), 2021. doi: <https://doi.org/10.1613/jair.1.12228>.
- Jane Elith, Steven J. Phillips, Trevor Hastie, Miroslav Dudík, Yung En Chee, and Colin J. Yates. A statistical explanation of MaxEnt for ecologists. *Diversity and Distributions*, 17(1):43–57, 1 2011. ISSN 13669516. doi: [10.1111/j.1472-4642.2010.00725.x](https://doi.org/10.1111/j.1472-4642.2010.00725.x).
- Tlameo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A Survey on Missing Data in Machine Learning. *Journal of Big Data*, 8(1), 12 2021. ISSN 21961115. doi: [10.1186/s40537-021-00516-9](https://doi.org/10.1186/s40537-021-00516-9).
- Council of the European Union. European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.
- Line Farah, Juliette M. Murris, Isabelle Borget, Agathe Guilloux, Nicolas M. Martelli, and Sandrine I.M. Katsahian. Assessment of Performance, Interpretability, and Explainability in Artificial Intelligence–Based Health Technologies: What Healthcare Stakeholders Need to Know. *Mayo Clinic Proceedings: Digital Health*, 1(2):120–138, 6 2023. ISSN 29497612. doi: [10.1016/j.mcpdig.2023.02.004](https://doi.org/10.1016/j.mcpdig.2023.02.004).
- M. A. H. Farquad, V. Ravi, and S. Bapi Raju. Rule Extraction from Support Vector Machine Using Modified Active Learning Based Approach: An Application to CRM. In *Knowledge-Based and Intelligent Information and Engineering Systems 14th International Conference (KES 2010)*, pages 461–470, 2010.

References

- Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *2nd International Conference on Knowledge Discovery and Data Mining*, 1996.
- Jacob Feldman. Minimization of Boolean complexity in human concept learning. *Nature*, 407 (6804):630–633, 10 2000. ISSN 00280836. doi: [10.1038/35036586](https://doi.org/10.1038/35036586).
- Montgomery Flora, Corey Potvin, Amy McGovern, and Shawn Handler. Comparing Explanation Methods for Traditional Machine Learning Models Part 1: An Overview of Current Methods and Quantifying Their Disagreement. *arXiv.org*, 11 2022a. doi: [10.48550/arXiv.2211.08943](https://doi.org/10.48550/arXiv.2211.08943).
- Montgomery Flora, Corey Potvin, Amy McGovern, and Shawn Handler. Comparing Explanation Methods for Traditional Machine Learning Models Part 2: Quantifying Model Explainability Faithfulness and Improvements with Dimensionality Reduction. *arXiv*, 11 2022b.
- Alex A Freitas. Comprehensible Classification Models-a position paper. *ACM SIGKDD Explorations*, 15(1):1–10, 2014.
- Paulo S.A. Freitas and António J.L. Rodrigues. Model combination in neural-based forecasting. *European Journal of Operational Research*, 173(3):801–814, 9 2006. ISSN 03772217. doi: [10.1016/j.ejor.2005.06.057](https://doi.org/10.1016/j.ejor.2005.06.057).
- Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 8 1997. ISSN 00220000. doi: [10.1006/jcss.1997.1504](https://doi.org/10.1006/jcss.1997.1504).
- Yoav Freund and Robert E Schapire. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.
- Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 9 2008. ISSN 1932-6157. doi: [10.1214/07-AOAS148](https://doi.org/10.1214/07-AOAS148).
- Drew Fudenberg, Jon Kleinberg, Annie Liang, and Sendhil Mullainathan. Measuring the Completeness of Economic Models. *Journal of Political Economy*, 130(4):956–990, 4 2022. ISSN 1537534X. doi: [10.1086/718371](https://doi.org/10.1086/718371).
- Johannes Fürnkranz. Pruning Algorithms for Rule Learning. *Machine Learning*, 27:139–172, 1997.
- Johannes Fürnkranz and Peter Flach. ROC 'n' Rule Learning-Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58:39–77, 2005. doi: [10.1007/s10994-005-5011-x](https://doi.org/10.1007/s10994-005-5011-x).
- Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Springer Science and Business Media, Germany, 2012.
- João Gama. *Combining Classification Algorithms*. PhD thesis, Faculdade de Ciências da Universidade do Porto, 1999.
- Carolina García-Ascanio and Carlos Maté. Electric power demand forecasting using interval time series: A comparison between VAR and iMLP. *Energy Policy*, 38(2):715–725, 2 2010. ISSN 03014215. doi: [10.1016/j.enpol.2009.10.007](https://doi.org/10.1016/j.enpol.2009.10.007).
- N. Garcia-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer. Cooperative Coevolution of Artificial Neural Network Ensembles for Pattern Classification. *IEEE Transactions on Evolutionary Computation*, 9(3):271–302, 6 2005. ISSN 1089-778X. doi: [10.1109/TEVC.2005.844158](https://doi.org/10.1109/TEVC.2005.844158).

References

- Hans U. Gerber, Bartholomew P.K. Leung, Elias S.W. Shiu, and Elias S.W. Shiu. Indicator function and hantendorff theorem. *North American Actuarial Journal*, 7(1):38–47, 1 2003. ISSN 10920277. doi: [10.1080/10920277.2003.10596075](https://doi.org/10.1080/10920277.2003.10596075).
- Philippe J. Giabbanelli and Joseph G. Peters. An Algebra to Merge Heterogeneous Classifiers, 1 2015.
- Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. In *International Conference on Data Science and Advanced Analytics*, 5 2018.
- Alan Keller Gomes. Small Disjuncts Grouping by Rule Coverage and Accuracy Measures. In *7th International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 412–415. IEEE, 2008. ISBN 0769529763. doi: [10.1109/ISDA.2007.112](https://doi.org/10.1109/ISDA.2007.112).
- Gloria González-Rivera and Javier Arroyo. Time series modeling of histogram-valued data: The daily histogram time series of S&P500 intradaily returns. *International Journal of Forecasting*, 28(1):20–33, 1 2012. ISSN 01692070. doi: [10.1016/j.ijforecast.2011.02.007](https://doi.org/10.1016/j.ijforecast.2011.02.007).
- Bryce Goodman and Seth Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, 6 2016. doi: [10.1609/aimag.v38i3.2741](https://doi.org/10.1609/aimag.v38i3.2741).
- K. Y. Gorbunov and V. A. Lyubetsky. The tree nearest on average to a given set of trees. *Problems of Information Transmission*, 47(3):274–288, 12 2011. ISSN 00329460. doi: [10.1134/S0032946011030069](https://doi.org/10.1134/S0032946011030069).
- K. Y. Gorbunov and V. A. Lyubetsky. Fast algorithm to reconstruct a species supertree from a set of protein trees. *Molecular Biology*, 46(1):161–167, 2 2012. ISSN 00268933. doi: [10.1134/S0026893312010086](https://doi.org/10.1134/S0026893312010086).
- John W. Graham. Missing Data Analysis: Making It Work in the Real World. *Annual Review of Psychology*, 60(1):549–576, 1 2009. ISSN 0066-4308. doi: [10.1146/annurev.psych.58.110405.085530](https://doi.org/10.1146/annurev.psych.58.110405.085530).
- Paul E. Green. A New Measure of Predictor Variable Importance in Multiple Regression. *Journal of Marketing Research*, 15:356, 1978.
- Crina Grosan and Ajith Abraham. Rule-Based Expert Systems. In *Intelligent Systems: A Modern Approach*, volume 17, pages 149–185. Springer-Verlag, Berlin Heidelberg, 2011. doi: [10.1007/978-3-642-21004-4_7](https://doi.org/10.1007/978-3-642-21004-4_7).
- Robert Grossman, Roberto. Bayardo, Kristin P. Bennett, Jaideep. Vaidya, Association for Computing Machinery. Special Interest Group on Knowledge Discovery & Data Mining., Association for Computing Machinery. Special Interest Group on Management of Data., and American Association for Artificial Intelligence. Rule extraction from linear support vector machines. In *11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 32–40. ACM Press, 2005. ISBN 159593135X.
- Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5):1–42, 2 2018. doi: [10.1145/3236009](https://doi.org/10.1145/3236009).

References

- Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Francesca Naretto, Franco Turini, Dino Pedreschi, and Fosca Giannotti. Stable and actionable explanations of black-box models through factual and counterfactual rules. *Data Mining and Knowledge Discovery*, 2022. ISSN 1573756X. doi: [10.1007/s10618-022-00878-5](https://doi.org/10.1007/s10618-022-00878-5).
- David Gunning, Eric Vorm, Jennifer Yunyan Wang, and Matt Turek. DARPA’s explainable AI (XAI) program: A retrospective. *Applied AI Letters*, 2(4):e61, 2021. doi: <https://doi.org/10.1002/ail2.61>.
- Lawrence O. Hall, Nitesh Chawla, and Kevin W. Bowyer. Combining Decision Trees Learned in Parallel. *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pages 10–15, 1998.
- Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 3rd edition, 2012.
- Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pages 3323–3331. Neural information processing systems foundation, 2016.
- J. A. Hartigan and M. A. Wong. A K-Means Clustering Algorithm. *Source: Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- Trevor Hastie, Robert Tibshirani Martin, and Wainwright Hastie. *Statistical Learning with Sparsity The Lasso and Generalizations Statistical Learning with Sparsity*. Chapman and Hall/CRC, 2015.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2017. doi: <https://doi.org/10.1007/978-0-387-84858-7>.
- Frederick Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, 9 1985. ISSN 0001-0782. doi: [10.1145/4284.4286](https://doi.org/10.1145/4284.4286).
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 770–778. IEEE Computer Society, 12 2016. ISBN 9781467388504. doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- Johan Holmén. *Navigation sustainability transformations: backcasting, transdisciplinarity and social learning*. PhD thesis, Chalmers University of Technology, 2020.
- Robert C Holte. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11:63–91, 1993. doi: [10.1023/A:1022631118932](https://doi.org/10.1023/A:1022631118932).
- Rima Houari, Ahcene Bounceur, A Kamel Tari, and M Tahar Kecha. Handling Missing Data Problems with Sampling Methods. In *2014 International Conference on Advanced Networking Distributed Systems and Applications*, pages 99–104. IEEE, 6 2014. doi: [10.1109/INDS.2014.25](https://doi.org/10.1109/INDS.2014.25).
- Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal Sparse Decision Trees. In *Conference on Neural Information Processing Systems (NeurIPS 2019)*, 4 2019.
- Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009. ISSN 00189448. doi: [10.1109/TIT.2009.2027527](https://doi.org/10.1109/TIT.2009.2027527).

References

- Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 4 2011. ISSN 01679236. doi: [10.1016/j.dss.2010.12.003](https://doi.org/10.1016/j.dss.2010.12.003).
- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R*. Springer, 2013. ISBN 9781461471370.
- Thomas J. Jech. *Set theory*. Springer-Verlag, Berlin Heidelberg, 2006. ISBN 0123819504.
- Ulf Johansson, Rikard König, and Lars Niklasson. The Truth is In There-Rule Extraction from Opaque Models Using Genetic Programming. In *Proc. FLAIRS Conf., 2004*, pages 658–663, 2004.
- Ulf Johansson, Tuve Löfström, Rikard König, Cecilia Sönströd, and Lars Niklasson. Rule Extraction from Opaque Models-A Slightly Different Perspective. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA'06)*, 2006.
- Kaggle. Kaggle datasets, 2024.
- Hyun Kang. The prevention and handling of the missing data. *Korean Journal of Anesthesiology*, 64(5):402–406, 5 2013. ISSN 20056419. doi: [10.4097/kjae.2013.64.5.402](https://doi.org/10.4097/kjae.2013.64.5.402).
- Aram Karalič and Ivan Bratko. First Order Regression. *Machine Learning*, 26:147–176, 1997.
- H. Kargupta and Byung-Hoon Park. A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):216–229, 2 2004. ISSN 1041-4347. doi: [10.1109/TKDE.2004.1269599](https://doi.org/10.1109/TKDE.2004.1269599).
- Been Kim, Caleb M. Chacha, and Julie A. Shah. Inferring Team Task Plans from Human Meetings: A Generative Modeling Approach with Logic-Based Prior. *Journal of Artificial Intelligence Research*, 52:361–398, 3 2015. ISSN 1076-9757. doi: [10.1613/jair.4496](https://doi.org/10.1613/jair.4496).
- Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not Enough, Learn to Criticize! Criticism for Interpretability. *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.
- Hyunjoong Kim and Wei-Yin Loh. Classification Trees With Unbiased Multiway Splits. *Journal of the American Statistical Association*, 96(454):589–604, 6 2001. ISSN 0162-1459. doi: [10.1198/016214501753168271](https://doi.org/10.1198/016214501753168271).
- Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 1137–1143, 1995.
- Ron Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
- Krzysztof J. Cios and Witold Pedrycz and Roman W. Swiniarski. *Data Mining: A Knowledge Discovery Approach*. Springer, 2007. ISBN 9780387355483.
- Max Kuhn. Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28(5), 2008. ISSN 1548-7660. doi: [10.18637/jss.v028.i05](https://doi.org/10.18637/jss.v028.i05).

References

- Max Kuhn and John Ross Quinlan. C50: C5.0 Decision Trees and Rule-Based Models, 2023.
- Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? Ways explanations impact end users’ mental models. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 3–10. IEEE, 9 2013. ISBN 978-1-4799-0369-6. doi: [10.1109/VLHCC.2013.6645235](https://doi.org/10.1109/VLHCC.2013.6645235).
- L.I. Kuncheva. Switching between selection and fusion in combining classifiers: an experiment. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 32(2):146–156, 4 2002. ISSN 10834419. doi: [10.1109/3477.990871](https://doi.org/10.1109/3477.990871).
- Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 1675–1684. Association for Computing Machinery, 8 2016. ISBN 9781450342322. doi: [10.1145/2939672.2939874](https://doi.org/10.1145/2939672.2939874).
- Niels Landwehr. Logistic Model Trees. *Machine Learning*, 59:161–205, 2005.
- Diane L. Langkamp, Amy Lehman, and Stanley Lemeshow. Techniques for Handling Missing Data in Secondary Analyses of Large Surveys. *Academic Pediatrics*, 10(3):205–210, 5 2010. ISSN 18762859. doi: [10.1016/j.acap.2010.01.005](https://doi.org/10.1016/j.acap.2010.01.005).
- Pat Langley and Herbert A. Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64, 11 1995. ISSN 0001-0782. doi: [10.1145/219717.219768](https://doi.org/10.1145/219717.219768).
- Nada Lavrač, Peter Flach, and Blaz Zupan. Rule Evaluation Measures: A Unifying View. In *Inductive Logic Programming*, pages 174–185. Springer, 1999. doi: [10.1007/3-540-48751-4_17](https://doi.org/10.1007/3-540-48751-4_17).
- Benjamin Leblanc and Pascal Germain. Interpretability in Machine Learning: on the Interplay with Explainability, Predictive Performances and Models. *arXiv*, 11 2023. doi: [10.48550/arXiv.2311.11491](https://doi.org/10.48550/arXiv.2311.11491).
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015. ISSN 0028-0836. doi: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- Antoni Ligeża. *Logical Foundations for Rule-Based Systems*, volume 11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-29117-6. doi: [10.1007/3-540-32446-1](https://doi.org/10.1007/3-540-32446-1).
- E. Lima, C. Mues, and B. Baesens. Domain knowledge integration in data mining using decision tables: Case studies in churn prediction. *Journal of the Operational Research Society*, 60(8): 1096–1106, 2009. ISSN 14769360. doi: [10.1057/jors.2008.161](https://doi.org/10.1057/jors.2008.161).
- Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and Scalable Optimal Sparse Decision Trees. In *37th International Conference on Machine Learning*, 6 2020.
- Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(18), 2021. doi: [10.3390/e23010018](https://doi.org/10.3390/e23010018).
- Rockard Lior and Oded Maimon. *Data Mining with Decision Trees: Theory and applications*. World Scientific, 2008.
- Zachary C. Lipton. The Mythos of Model Interpretability. *Queue*, 16(3):31–57, 6 2018.

References

- P. J.G. Lisboa, S. Saralajew, A. Vellido, R. Fernández-Domenech, and T. Villmann. The coming of age of interpretable and explainable machine learning models. *Neurocomputing*, 535:25–39, 5 2023. ISSN 18728286. doi: [10.1016/j.neucom.2023.02.040](https://doi.org/10.1016/j.neucom.2023.02.040).
- D. Lucy and R. Aykroyd. GenKern: Functions for generating and manipulating binned kernel density estimates, 2010.
- Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions, 5 2017.
- Fengqiao Luo and Sanjay Mehrotra. A Concentration Result of Estimating Phi-Divergence using Data Dependent Partition, 1 2018.
- O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, 2nd edition, 2010.
- K. V. Mardia. Statistics of Directional Data. *Source: Journal of the Royal Statistical Society. Series B (Methodological)*, 37(3):349–393, 1975.
- Vincent Margot and George Luta. A New Method to Compare the Interpretability of Rule-Based Algorithms. *AI*, 2(4):621–635, 11 2021. doi: [10.3390/ai2040037](https://doi.org/10.3390/ai2040037).
- Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. UCI Machine Learning Repository, 2023.
- David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011. ISSN 01679236. doi: [10.1016/j.dss.2011.01.013](https://doi.org/10.1016/j.dss.2011.01.013).
- Nicolai Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4(4), 12 2010. ISSN 1932-6157. doi: [10.1214/10-AOAS367](https://doi.org/10.1214/10-AOAS367).
- Jerry M. Mendel. *Uncertain Rule-Based Fuzzy Systems*. Springer International Publishing, Cham, 2nd edition, 2017. ISBN 978-3-319-51369-0. doi: [10.1007/978-3-319-51370-6](https://doi.org/10.1007/978-3-319-51370-6).
- João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression. *ACM Computing Surveys*, 45(1):1–40, 11 2012. ISSN 0360-0300. doi: [10.1145/2379776.2379786](https://doi.org/10.1145/2379776.2379786).
- Christopher Merz. *Classification and Regression by Combining Models*. PhD thesis, University of California, 1998.
- Andreas Messalas, Yiannis Kanellopoulos, and Christos Makris. Model-Agnostic Interpretability with Shapley Values. In *10th International Conference on Information, Intelligence, Systems and Applications, IISA 2019*. Institute of Electrical and Electronics Engineers Inc., 7 2019. ISBN 9781728149592. doi: [10.1109/IISA.2019.8900669](https://doi.org/10.1109/IISA.2019.8900669).
- Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence*, 6 2017. doi: [10.1016/J.ARTINT.2018.07.007](https://doi.org/10.1016/J.ARTINT.2018.07.007).
- Yao Ming, Huamin Qu, and Enrico Bertini. RuleMatrix: Visualizing and Understanding Classifiers with Rules. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):342–352, 1 2019. ISSN 1077-2626. doi: [10.1109/TVCG.2018.2864812](https://doi.org/10.1109/TVCG.2018.2864812).
- Tom Michael Mitchell. *Machine Learning*. McGraw-Hil, 1997. ISBN 0070428077.

References

- Atsushi Mizumoto. Calculating the Relative Importance of Multiple Regression Predictor Variables Using Dominance Analysis and Random Forests. *Language Learning*, 73(1):161–196, 3 2023. ISSN 0023-8333. doi: [10.1111/lang.12518](https://doi.org/10.1111/lang.12518).
- Annette M. Molinaro, Richard Simon, and Ruth M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 8 2005. ISSN 1367-4803. doi: [10.1093/bioinformatics/bti499](https://doi.org/10.1093/bioinformatics/bti499).
- Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. Leanpub, 2019.
- Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014. ISSN 01679236. doi: [10.1016/j.dss.2014.03.001](https://doi.org/10.1016/j.dss.2014.03.001).
- W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 10 2019. ISSN 0027-8424. doi: [10.1073/pnas.1900654116](https://doi.org/10.1073/pnas.1900654116).
- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 9780262018029.
- Grzegorz J. Nalepa. *Modeling with Rules Using Semantic Knowledge Engineering*, volume 130. Springer International Publishing, Cham, 2018. ISBN 978-3-319-66654-9. doi: [10.1007/978-3-319-66655-6](https://doi.org/10.1007/978-3-319-66655-6).
- Nekuri Naveen, V. Ravi, and C. Raghavendra Rao. Data Mining via Rules Extracted from GMDH: An Application to Predict Churn in Bank Credit Cards. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2010)*, pages 80–89. Springer, Berlin, 2010. doi: [10.1007/978-3-642-15387-7_12](https://doi.org/10.1007/978-3-642-15387-7_12).
- Josue Obregon, Aekyung Kim, and Jae Yoon Jung. RuleCOSI: Combination and simplification of production rules from boosted decision trees for imbalanced classification. *Expert Systems with Applications*, 126:64–82, 7 2019. ISSN 09574174. doi: [10.1016/j.eswa.2019.02.012](https://doi.org/10.1016/j.eswa.2019.02.012).
- David Opitz and Richard Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
- Fernando E. B. Otero and Alex A. Freitas. Improving the Interpretability of Classification Rules. In *Proc. 2013 Genetic and Evolutionary Computation Conference (GECCO’13)*, page 1640. ACM, 2013. ISBN 9781450319638.
- Fabian Pedregosa, Vincent Michel, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, David Cournapeau, Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Alexandre Passos, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Witold Pedrycz and Fernando Gomide. *Fuzzy systems engineering: Toward human-centric computing*. John Wiley Sons, 2007.

References

- Emmanuel Périnel and Yves Lechevallier. Symbolic Discrimination Rules. *Analysis of Symbolic Data, Exploratory Methods for Extracting Statistical Information from Complex Data*, pages 244–265, 2000.
- Petar Petrovski and Christian Bizer. Learning Expressive Linkage Rules from Sparse Data. *Semantic Web*, 0:1–1, 2016.
- Richard R. Picard and R. Dennis Cook. Cross-Validation of Regression Models. *Journal of the American Statistical Association*, 79(387):575–583, 9 1984. ISSN 0162-1459. doi: [10.1080/01621459.1984.10478083](https://doi.org/10.1080/01621459.1984.10478083).
- T. Piketty and E. Saez. Income Inequality in the United States, 1913-1998. *The Quarterly Journal of Economics*, 118(1):1–41, 2 2003. ISSN 0033-5533. doi: [10.1162/00335530360535135](https://doi.org/10.1162/00335530360535135).
- Ghanshyam Pilia, Prasanna V. Balachandran, James E. Gubernatis, and Turab Lookman. Multi-Fidelity Learning. In *Data-Based Methods for Materials Design and Discovery. Synthesis Lectures on Materials and Optics*, pages 135–152. Springer, Cham., 2020. doi: [10.1007/978-3-031-02383-5_6](https://doi.org/10.1007/978-3-031-02383-5_6).
- R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6 (3):21–45, 2006. ISSN 1531-636X. doi: [10.1109/MCAS.2006.1688199](https://doi.org/10.1109/MCAS.2006.1688199).
- Nico Potyka, Xiang Yin, and Francesca Toni. On the Tradeoff Between Correctness and Completeness in Argumentative Explainable AI, 2022.
- Foster John Provost and Daniel N Hennessy. Scaling Up: Distributed Machine Learning with Cooperation. *13th National Conference on Artificial Intelligence (AAAI’96)*, pages 74–79, 1996.
- Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):85, 12 2016. ISSN 1687-6180. doi: [10.1186/s13634-016-0382-7](https://doi.org/10.1186/s13634-016-0382-7).
- John Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.
- John Ross Quinlan. Generating Production Rules From Decision Trees. *ijcai*, 87:307–307, 1987a.
- John Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987b.
- John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.
- R Core Team. R: A Language and Environment for Statistical Computing, 2023.
- Michael Rapp, Eneldo Loza Mencía, and Johannes Fürnkranz. On the Trade-Off Between Consistency and Coverage in Multi-label Rule Learning Heuristics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11828 LNAI(Mlc):96–111, 2019. ISSN 16113349. doi: [10.1007/978-3-030-33778-0_9](https://doi.org/10.1007/978-3-030-33778-0_9).
- Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, S. M. Ali Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions. In *6th International Conference on Learning Representations (ICLR 2018)*, 10 2018.

References

- Alvin C. Rencher and G. Bruce. Schaalje. *Linear models in statistics*. John Wiley & Sons, 2008. ISBN 9780471754985.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016:1135–1144, 8 2016. doi: [10.1145/2939672.2939778](https://doi.org/10.1145/2939672.2939778).
- Ronald L Rivest. Learning Decision Lists. *Machine Learning*, 2:229–246, 1987.
- Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2 2010. ISSN 0269-2821. doi: [10.1007/s10462-009-9124-7](https://doi.org/10.1007/s10462-009-9124-7).
- Andrea Romei and Salvatore Ruggieri. A multidisciplinary survey on discrimination analysis, 4 2013. ISSN 14698005.
- Niall Rooney, David Patterson, Sarab Anand, and Alexey Tsymbal. Dynamic Integration of Regression Models. In *International Workshop on Multiple Classifier Systems*, pages 164–173, 2004. doi: [10.1007/978-3-540-25966-4_16](https://doi.org/10.1007/978-3-540-25966-4_16).
- Kenneth H. Rosen. *Discrete Mathematics and Its Applications (7th Edition)*. McGraw Hill, 2012. ISBN 9780073383095.
- Jennifer Rowley. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33(2):163–180, 2007. doi: [10.1177/0165551506070706](https://doi.org/10.1177/0165551506070706).
- Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 5 2019. ISSN 2522-5839. doi: [10.1038/s42256-019-0048-x](https://doi.org/10.1038/s42256-019-0048-x).
- David Rumelhart, Geoff Hinton, and Ronald Williams. Backpropagation: A Learning Algorithm for Neural Networks. *Nature*, 323:533–536, 1986.
- Stefan Rüping. Learning with Local Models. *Local Pattern Detection*, pages 153–170, 2005.
- Stefan Rüping. *Learning interpretable models*. PhD thesis, Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, 2006.
- A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 7 1959. ISSN 0018-8646. doi: [10.1147/rd.33.0210](https://doi.org/10.1147/rd.33.0210).
- R. Sarker, H. Abbass, and C. Newton. *Heuristics and Optimization for Knowledge Discovery*. IGI Global, 2000. doi: [10.4018/9781930708266.ch001](https://doi.org/10.4018/9781930708266.ch001).
- Patrick Schober, Christa Boer, and Lothar A. Schwarte. Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, 126(5):1763–1768, 5 2018. ISSN 0003-2999. doi: [10.1213/ANE.0000000000002864](https://doi.org/10.1213/ANE.0000000000002864).
- David W Scott. Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions. *Source: The Annals of Statistics*, 13(3):1024–1040, 1985.
- David W. Scott. Multivariate Density Estimation and Visualization, 2004.
- David W. Scott. Averaged shifted histogram. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):160–164, 3 2010. ISSN 19395108. doi: [10.1002/wics.54](https://doi.org/10.1002/wics.54).

References

- Sharath M. Shankaranarayana and Davor Runje. ALIME: Autoencoder Based Approach for Local Interpretability. In Hujun Yin, David Camacho, Peter Tino, Antonio J. Tallón-Ballesteros, Ronaldo Menezes, and Richard Allmendinger, editors, *International Conference on Intelligent Data Engineering and Automated Learning*, volume 11871, pages 454–463, Cham, 2019. Springer International Publishing. doi: [10.1007/978-3-030-33607-3_49](https://doi.org/10.1007/978-3-030-33607-3_49).
- William D. Shannon and David Banks. Combining classification trees using MLE. *Statistics in Medicine*, 18(6):727–740, 3 1999.
- Xiaoling Shu and Yiwan Ye. Knowledge Discovery: Methods from data mining and machine learning. *Social Science Research*, 110:102817, 2 2023. ISSN 0049089X. doi: [10.1016/j.ssresearch.2022.102817](https://doi.org/10.1016/j.ssresearch.2022.102817).
- Wilson Silva, Kelwin Fernandes, Maria J. Cardoso, and Jaime S. Cardoso. Towards complementary explanations using Deep Neural Networks. In Danail Stoyanov, Zeike Taylor, Seyed Mostafa Kia, Ipek Oguz, Mauricio Reyes, Anne Martel, Lena Maier-Hein, Andre F. Marquand, Edouard Duchesnay, Tommy Löfstedt, Bennett Landman, M. Jorge Cardoso, Carlos A. Silva, Sergio Pereira, and Raphael Meier, editors, *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, volume 11038 of *Lecture Notes in Computer Science*, pages 133–140, Cham, 2018. Springer International Publishing. ISBN 978-3-030-02627-1. doi: [10.1007/978-3-030-02628-8](https://doi.org/10.1007/978-3-030-02628-8).
- B. W. Silverman. Density Estimation for Statistics and Data Analysis. In *Monographs on Statistics and Applied Probability*. Chapman and Hall, 1986. ISBN 0412246201.
- Louisa H. Smith and Tyler J. Vanderweele. Bounding Bias Due to Selection. *Epidemiology*, 30(4):509–516, 7 2019. ISSN 15315487. doi: [10.1097/EDE.0000000000001032](https://doi.org/10.1097/EDE.0000000000001032).
- Michael Smithson and Yiyun Shou. How Big Is (Sample) Space? Judgment and Decision Making With Unknown States and Outcomes. *Decision*, 8(4):237–256, 2021. ISSN 23259973. doi: [10.1037/dec0000154](https://doi.org/10.1037/dec0000154).
- Jan Tobias Sohns, Christoph Garth, and Heike Leitte. Decision Boundary Visualization for Counterfactual Reasoning. *Computer Graphics Forum*, 2022. ISSN 14678659. doi: [10.1111/cgf.14650](https://doi.org/10.1111/cgf.14650).
- Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, 7 2009. ISSN 03064573. doi: [10.1016/j.ipm.2009.03.002](https://doi.org/10.1016/j.ipm.2009.03.002).
- Fabio Somenzi. Binary Decision Diagrams. Technical report, University of Colorado, 1999.
- Edgar Sommer. Theory Restructuring: Coarse-grained Integration of Strategies for Induction & Maintenance of Knowledge Bases. In *Proceedings of the Third International Conference on Multistrategy Learning*, 1996.
- Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Information Science and Statistics. Springer New York, New York, NY, 2008. ISBN 978-0-387-77241-7. doi: [10.1007/978-0-387-77242-4](https://doi.org/10.1007/978-0-387-77242-4).
- Leon S. Sterling and Ehud Y. Shapiro. *The Art of Prolog, Advanced Programming Techniques*. The MIT Press, 1994. ISBN 9780262691635.

References

- M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1 1974. ISSN 0035-9246. doi: [10.1111/j.2517-6161.1974.tb00994.x](https://doi.org/10.1111/j.2517-6161.1974.tb00994.x).
- Pedro Strecht. A Survey of Merging Decision Trees Data Mining Approaches. In A. A. Sousa and E. Oliveira, editors, *10th Doctoral Symposium in Informatics Engineering (DSIE'15)*, pages 36–47, 2015. doi: [10.13140/2.1.3987.4880](https://doi.org/10.13140/2.1.3987.4880).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Educational Data Mining: Preliminary Results at University of Porto. *EUNIS Journal of Higher Education IT Issue*, 1(2015), 2014a. doi: [10.13140/2.1.3463.2009](https://doi.org/10.13140/2.1.3463.2009).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Merging Decision Trees: A Case Study in Predicting Student Performance. In *Advanced Data Mining and Applications. ADMA 2014. Lecture Notes in Computer Science*, volume 8933, pages 535–548. Springer International Publishing, 2014b. doi: [10.1007/978-3-319-14717-8_42](https://doi.org/10.1007/978-3-319-14717-8_42).
- Pedro Strecht, Luís Cruz, Carlos Soares, João Mendes-Moreira, and Rui Abreu. A Comparative Study of Classification and Regression Algorithms for Modelling Students' Academic Performance. In *8th International Conference on Educational Data Mining*, pages 392–395, 2015a. URL <https://eric.ed.gov/?id=ED560769>.
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Exploring the Impact of Ordering Models in Merging Decision Trees: A Case Study in Education. In *Doctoral Consortium, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2015)*, pages 216–225, 2015b. URL <https://www.researchgate.net/publication/283326005>.
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. A Framework for Analytical Approaches to Combine Interpretable Models. In *Information Management and Big Data (SIMBig 2018). Communications in Computer and Information Science*, volume 898 of *Communications in Computer and Information Science*, pages 182–197. Springer, Cham, 2018. doi: [10.1007/978-3-030-11680-4_19](https://doi.org/10.1007/978-3-030-11680-4_19).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Generalizing Knowledge in Decentralized Rule-Based Models. In A. Monreale, C. Alzate, M. Kamp, Y. Krishnamurthy, D. Paurat, M. Sayed-Mouchaweh, A. Bifet, J. Gama, and R.P. Ribeiro, editors, *Decentralized Machine Learning on the Edge Workshops. ECML PKDD 2018. Communications in Computer and Information Science*, volume 967, pages 29–36. Springer, Cham, 2019. doi: [10.1007/978-3-030-14880-5_3](https://doi.org/10.1007/978-3-030-14880-5_3).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Inmplode: A framework to interpret multiple related rule-based models. *Expert Systems*, 38(6):e12702, 9 2021. ISSN 0266-4720. doi: [10.1111/exsy.12702](https://doi.org/10.1111/exsy.12702).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Density Estimation in High-Dimensional Spaces: A Multivariate Histogram Approach. In *Advanced Data Mining and Applications. ADMA 2022. Lecture Notes in Computer Science*, volume 13726, pages 266–278. Springer, Cham, 2022. doi: [10.1007/978-3-031-22137-8_20](https://doi.org/10.1007/978-3-031-22137-8_20).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Symbolic data analysis to improve completeness of model combination methods. In *Advances in Artificial Intelligence, AI 2023*.

References

- Lecture Notes in Computer Science*, volume 14472, pages 107–119. Springer, 2024. doi: [10.1007/978-981-99-8391-9_9](https://doi.org/10.1007/978-981-99-8391-9_9).
- Pedro Strecht, João Mendes-Moreira, and Carlos Soares. Estimating completeness of consensus models: Geometrical and distributional approaches. In Giuseppe Nicosia, Panos Pardalos, and Varun Ojha, editors, *The 10th International Conference on machine Learning, Optimization and Data science, LOD 2024, Lecture Notes in Computer Science*, volume 15508, pages 464–478. Springer, 2025. doi: [10.1007/978-3-031-82481-4_32](https://doi.org/10.1007/978-3-031-82481-4_32).
- Guolong Su, Dennis Wei, Kush R. Varshney, and Dmitry M. Malioutov. Interpretable Two-level Boolean Rule Learning for Classification. In *2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, 6 2016. doi: [10.48550/arXiv.1606.05798](https://doi.org/10.48550/arXiv.1606.05798).
- Motoi Suwa, A. Carlisle Scott, and Eduard H. Shortliffe. An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System. *AI Magazine*, 3(4):16–21, 1982.
- Pang-Nin Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India., 2006. ISBN 9781292026152.
- Madjid Tavana and Vahid Hajipour. A practical review and taxonomy of fuzzy expert systems: methods and applications. *Benchmarking: An International Journal*, 27(1):81–136, 9 2019. ISSN 1463-5771. doi: [10.1108/BIJ-04-2019-0178](https://doi.org/10.1108/BIJ-04-2019-0178).
- The Royal Society. *Explainable AI: the basics (Policy briefing)*. The Royal Society, 2019. ISBN 978-1-78252-433-5.
- Terry Therneau, Beth Atkinson, and Brian Ripley. rpart: Recursive Partitioning and Regression Trees, 2023.
- K Thulasiraman and M N S Swamy. *Graphs: Theory and Algorithms*. John Wiley & Sons, Inc., USA, 1992. ISBN 0471513563.
- Weida Tong, Huixiao Hong, Hong Fang, Qian Xie, and Roger G. Perkins. Decision Forest: Combining the Predictions of Multiple Independent Decision Tree Models. *Journal of Chemical Information and Computer Sciences*, 43(2):525–531, 2003.
- Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, 3 2016. ISSN 0885-6125. doi: [10.1007/s10994-015-5528-6](https://doi.org/10.1007/s10994-015-5528-6).
- Daan Van Der Valk and Stjepan Picek. Bias-variance Decomposition in Machine Learning-based Side-channel Analysis. *Cryptology ePrint Archive*, 2019.
- Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning, 7 2014.
- J. Vanthienen and G. Wets. From decision tables to expert system shells. *Data & Knowledge Engineering*, 13(3):265–282, 1994. doi: [10.1016/0169-023X\(94\)00020-4](https://doi.org/10.1016/0169-023X(94)00020-4).
- Alexandros Vassiliades, Nick Bassiliades, and Theodore Patkos. Argumentation and explainable artificial intelligence: A survey. *Knowledge Engineering Review*, 36, 2021. ISSN 14698005. doi: [10.1017/S026988921000011](https://doi.org/10.1017/S026988921000011).

References

- Mythreyi Velmurugan, Chun Ouyang, Catarina Moreira, and Renuka Sindhgatta. Developing a Fidelity Evaluation Approach for Interpretable Machine Learning. *arXiv*, 6 2021.
- Wouter Verbeke, David Martens, Christophe Mues, and Bart Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354–2364, 3 2011. ISSN 09574174. doi: [10.1016/j.eswa.2010.08.023](https://doi.org/10.1016/j.eswa.2010.08.023).
- Anca Vermesan and Frans Coenen. *Validation and Verification of Knowledge Based Systems*. Springer US, Boston, MA, 1999. ISBN 978-1-4419-5107-6. doi: [10.1007/978-1-4419-5107-6](https://doi.org/10.1007/978-1-4419-5107-6).
- Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation. *SSRN Electronic Journal*, 2016. ISSN 1556-5068. doi: [10.2139/ssrn.2903469](https://doi.org/10.2139/ssrn.2903469).
- M. Wand and B. Ripley. KernSmooth: Functions for kernel smoothing for Wand & Jones (1995), 2010.
- Stanisław Węglarczyk. Kernel density estimation and its application. *ITM Web of Conferences*, 23:00037, 2018.
- Sholom M. Weiss and Nitin Indurkha. Optimized Rule Induction. *IEEE Expert-Intelligent Systems and their Applications*, 8(6):61–69, 1993. ISSN 08859000. doi: [10.1109/64.248354](https://doi.org/10.1109/64.248354).
- Sholom M. Weiss and Nitin Indurkha. Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.
- Graham J. Williams. Combining Decision Trees. *Artificial Intelligence Developments and Applications*, pages 273–289, 1988.
- Ian Witten. Inducing Model Trees for Continuous Classes. Technical report, University of Waikato, 1997.
- Dirk U. Wulff, Max Mergenthaler-Canseco, and Ralph Hertwig. A meta-analytic review of two modes of learning and the description-experience gap. *Psychological Bulletin*, 144(2):140–176, 2 2018. ISSN 00332909. doi: [10.1037/bul0000115](https://doi.org/10.1037/bul0000115).
- Ziqi Yang. Fidelity: A Property of Deep Neural Networks to Measure the Trustworthiness of Prediction Results. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 676–678, New York, NY, USA, 7 2019. ACM. ISBN 9781450367523. doi: [10.1145/3321705.3331005](https://doi.org/10.1145/3321705.3331005).
- Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On Completeness-aware Concept-Based Explanations in Deep Neural Networks. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20554–20565. Curran Associates, Inc., 2020.
- Xue Ying. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168:022022, 2 2019. ISSN 1742-6588. doi: [10.1088/1742-6596/1168/2/022022](https://doi.org/10.1088/1742-6596/1168/2/022022).
- L. A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11(1-3):199–227, 1983. ISSN 01650114. doi: [10.1016/S0165-0114\(83\)80081-5](https://doi.org/10.1016/S0165-0114(83)80081-5).

References

- Yi Zhong, Jianghua He, and Prabhakar Chalise. Nested and Repeated Cross Validation for Classification Model With High-Dimensional Data. *Revista Colombiana de Estadística*, 43(1):103–125, 1 2020. ISSN 2389-8976. doi: [10.15446/rce.v43n1.80000](https://doi.org/10.15446/rce.v43n1.80000).
- Zhi-Hua Zhou. Rule Extraction: Using Neural Networks or for Neural Networks? *Journal of Computer Science and Technology*, 19(2):249–253, 2004.

References